```
 1: ********************************************************************************
 2:  Constructor Function and Procedure List of maXbox 3.9.9
 3: ********************************************************************************
 4:
 5: ////////////////////////////////////////////////////////////////////////////////
 6: ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX
 7: --------------------------------------------------------------------------------
 8:
 9: File Size of EXE: 22620672 V3.9.9.98 August 2014 To EKON/BASTA/JAX/IBZ/SWS/EU/DD
10: ***************Now the Funclist*****************************************
11: Funclist Function : 13140 //10766 //10165 (7648)
12: ***************Now the Proclist*****************************************
13: Proclist Procedure Size is: 8079  //6792 //6401 4752
14: ***************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1301 //995 //
16: def head:max: maXbox7: 11.08.2014 19:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: --------------------------------------------------------------------------------
20: Funclist total Size all is: 22520! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 22284
22: ASize of EXE: 22620672 (22041600) (16586240) (13511680)
23: SHA1 Hash of maXbox 3.9.9.98: 4C7A84045994D69DA0131D3E518D9B4901F20C5D
24:
25: --------------------------------------------------------------------------------
26: --------------------------------------------------------------------------------
27: ////////////////////////////////////////////////////////////////////////////////
28:
29:
30: FUNCTION Metric of Script: 256_findfunctions2_of_EXE.txt
31: Function *************Now the Funclist*****************
32: function  GetResStringChecked(Ident: string; const Args: array of const): string
33: Function ( Index : Longint) : Integer
34: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
35: Function _CheckAutoResult( ResultCode : HResult) : HResult
36: function _T(Name: tbtString): Variant;
37: function ABNFToText(const AText : String) : String
38: Function Abs(e : Extended) : Extended;
39: Function Ackermann( const A, B : Integer) : Integer
40: Function AcquireLayoutLock : Boolean
41: Function ActionByName( const AName : string) : TWebActionItem
42: Function ACTIVEBUFFER : PCHAR
43: Function Add : TAggregate
44: function Add : TCollectionItem
45: Function Add : TColumn
46: Function Add : TComboExItem
47: Function Add : TCookie
48: Function Add : TCoolBand
49: Function Add : TFavoriteLinkItem
50: Function Add : TFileTypeItem
51: Function Add : THeaderSection
52: Function Add : THTMLTableColumn
53: Function Add : TIdEMailAddressItem
54: Function Add : TIdMessagePart
55: Function Add : TIdUserAccount
56: Function Add : TListColumn
57: Function Add : TListItem
58: Function Add : TStatusPanel
59: Function Add : TTaskDialogBaseButtonItem
60: Function Add : TWebActionItem
61: Function Add : TWorkArea
62: Function Add( AClass : TClass) : Integer
63: Function Add( AComponent : TComponent) : Integer
64: Function Add( AItem, AData : Integer) : Integer
65: Function Add( AItem, AData : Pointer) : Pointer
66: Function Add( AItem, AData : TObject) : TObject
67: Function Add( AObject : TObject) : Integer
68: Function Add( const Access, Count : Cardinal; const Offset : Int64) : Integer
69: Function Add( const S : WideString) : Integer
70: Function Add( Image, Mask : TBitmap) : Integer
71: Function Add( Index : LongInt; const Text : string) : LongInt
72: Function Add( Sibling : TTreeNode; const S : string) : TTreeNode
73: Function Add(const S: string): Integer
74: function Add(S: string): Integer;
75: Function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer) : Integer
76: Function ADDCHILD : TFIELDDEF
77: Function AddChild( Index : LongInt; const Text : string) : LongInt
78: Function AddChild( Parent : TTreeNode; const S : string) : TTreeNode
79: Function AddChildFirst( Parent : TTreeNode; const S : string) : TTreeNode
80: Function AddChildObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
81: Function AddChildObject( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
82: Function AddChildObjectFirst( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
83: Function ADDFIELDDEF : TFIELDDEF
84: Function AddFileExtIfNecessary( AFileName, AExt : string) : string
85: Function AddFirst( Sibling : TTreeNode; const S : string) : TTreeNode
86: Function AddIcon( Image : TIcon) : Integer
87: Function AddImage( Value : TCustomImageList; Index : Integer) : Integer
88: Function ADDINDEXDEF : TINDEXDEF
89: Function AddItem(const Caption:String;const ImageIdx,SelectImageIdx,OverlayImagIdx,
    Indent:Int;Data:Ptr):TComboExItem
```

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

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

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

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

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

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

```
613: Function DegToRad2( const Value : Single) : Single;
614: Function DelChar( const pStr : string; const pChar : Char) : string
615: Function DelEnvironmentVar( const Name : string) : Boolean
616: Function Delete( const MsgNum : Integer) : Boolean
617: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
618: Function DeleteFile(const FileName: string): boolean)
619: Function DeleteFileEx( FileName : string; Flags : FILEOP_FLAGS) : Boolean
620: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
621: Function DelimiterPosn1(const sString:string;const sDelimiters:string;out cDelimiter: char) : integer;
622: Function DelSpace( const pStr : string) : string
623: Function DelString( const pStr, pDelStr : string) : string
624: Function DelTree( const Path : string) : Boolean
625: Function Depth : Integer
626: Function Description : string
627: Function DescriptionsAvailable : Boolean
628: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily) : Boolean
629: Function DescriptionToConvType( const ADescription : string; out AType : TConvType) : Boolean;
630: Function DescriptionToConvType1(const AFamil:TConvFamily;const ADescr:string;out AType:TConvType):Boolean;
631: Function DetectUTF8Encoding( const s : UTF8String) : TEncodeType
632: Function DialogsToPixelsX( const Dialogs : Word) : Word
633: Function DialogsToPixelsY( const Dialogs : Word) : Word
634: Function Digits( const X : Cardinal) : Integer
635: Function DirectoryExists( const Name : string) : Boolean
636: Function DirectoryExists( Directory : string) : Boolean
637: Function DiskFree( Drive : Byte) : Int64
638: function DiskFree(Drive: Byte): Int64)
639: Function DiskInDrive( Drive : Char) : Boolean
640: Function DiskSize( Drive : Byte) : Int64
641: function DiskSize(Drive: Byte): Int64)
642: Function DISPATCHCOMMAND( ACOMMAND : WORD) : BOOLEAN
643: Function DispatchEnabled : Boolean
644: Function DispatchMask : TMask
645: Function DispatchMethodType : TMethodType
646: Function DISPATCHPOPUP( AHANDLE : HMENU) : BOOLEAN
647: Function DispatchRequest( Sender : TObject; Request : TWebRequest; Response : TWebResponse) : Boolean
648: Function DisplayCase( const S : String) : String
649: Function DisplayRect( Code : TDisplayCode) : TRect
650: Function DisplayRect( TextOnly : Boolean) : TRect
651: Function DisplayStream( Stream : TStream) : string
652: TBufferCoord', 'record Char : integer; Line : integer; end
653: TDisplayCoord', 'record Column : integer; Row : integer; end
654: Function DisplayCoord( AColumn, ARow : Integer) : TDisplayCoord
655: Function BufferCoord( AChar, ALine : Integer) : TBufferCoord
656: Function DomainName( const AHost : String) : String
657: Function DosPathToUnixPath( const Path : string) : string
658: Function DottedLineTo( const Canvas : TCanvas; const X, Y : Integer) : Boolean;
659: Function DoubleDecliningBalance( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
660: Function DoubleToBcd( const AValue : Double) : TBcd;
661: Function DoubleToHex( const D : Double) : string
662: Function DoUpdates : Boolean
663: Function Dragging: Boolean;
664: Function DrawCaption( p1 : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL
665: Function DrawAnimatedRects( hwnd : HWND; idAni : Integer; const lprcFrom, lprcTo : TRect) : BOOL
666: Function DrawEdge( hdc : HDC; var qrc : TRect; edge : UINT; grfFlags : UINT) : BOOL
667: Function DrawFrameControl( DC : HDC; const Rect : TRect; uType, uState : UINT) : BOOL
668: {Works like InputQuery but displays 2edits. If PasswordChar <> #0, the second edit's PasswordChar is set}
669: Function DualInputQuery(const ACapt,Prpt1,Prpt2:string;var AVal1,AVal2:string;PasswrdChar:Char= #0):Bool;
670: Function DupeString( const AText : string; ACount : Integer) : string
671: Function Edit : Boolean
672: Function EditCaption : Boolean
673: Function EditText : Boolean
674: Function EditFolderList( Folders : TStrings) : Boolean
675: Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
676: Function Elapsed( const Update : Boolean) : Cardinal
677: Function EnableProcessPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
678: Function EnableThreadPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
679: Function EncodeDate( Year, Month, Day : Word) : TDateTime
680: function EncodeDate(Year, Month, Day: Word): TDateTime;
681: Function EncodeDateDay( const AYear, ADayOfYear : Word) : TDateTime
682: Function EncodeDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : TDateTime
683: Function EncodeDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSecond: Word): TDateTime
684: Function EncodeDateWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
685: Function EncodeDayOfWeekInMonth( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word) : TDateTime
686: Function EncodeString( s : string) : string
687: Function DecodeString( s : string) : string
688: Function EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
689: function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
690: Function EndIP : String
691: Function EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
692: Function EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
693: Function EndOfAMonth( const AYear, AMonth : Word) : TDateTime
694: Function EndOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
695: Function EndOfAYear( const AYear : Word) : TDateTime
696: Function EndOfTheDay( const AValue : TDateTime) : TDateTime
697: Function EndOfTheMonth( const AValue : TDateTime) : TDateTime
698: Function EndOfTheWeek( const AValue : TDateTime) : TDateTime
699: Function EndOfTheYear( const AValue : TDateTime) : TDateTime
700: Function EndPeriod( const Period : Cardinal) : Boolean
701: Function EndsStr( const ASubText, AText : string) : Boolean
```

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

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

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

```
 963: FUNCTION FormatInt64(i: int64): STRING;
 964: Function FormatMaskText( const EditMask : string; const Value : string) : string
 965: Function FormatValue( AValue : Cardinal) : string
 966: Function FormatVersionString( const HiV, LoV : Word) : string;
 967: Function FormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
 968: function Frac(X: Extended): Extended);
 969: Function FreeResource( ResData : HGLOBAL) : LongBool
 970: Function FromCommon( const AValue : Double) : Double
 971: function FromCommon(const AValue: Double): Double;
 972: Function FTPGMTDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 973: Function FTPLocalDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 974: Function FTPMLSToGMTDateTime( const ATimeStamp : String) : TDateTime
 975: Function FTPMLSToLocalDateTime( const ATimeStamp : String) : TDateTime
 976: Function FuncIn(AValue: Variant; ASet: Variant): Boolean;
 977: //Function Funclist Size is: 6444 of mX3.9.8.9
 978: Function FutureValue(const Rate:Extended;NPeriods:Integer;const Payment,PresentValue:Extended;PaymentTime:
      TPaymentTime):Extended
 979: Function Gauss( const x, Spread : Double) : Double
 980: function Gauss(const x,Spread: Double): Double;
 981: Function GCD(x, y : LongInt) : LongInt;
 982: Function GCDJ( X, Y : Cardinal) : Cardinal
 983: Function GDAL: LongWord
 984: Function GdiFlush : BOOL
 985: Function GdiSetBatchLimit( Limit : DWORD) : DWORD
 986: Function GdiGetBatchLimit : DWORD
 987: Function GenerateHeader : TIdHeaderList
 988: Function GeometricMean( const X : TDynFloatArray) : Float
 989: Function Get( AURL : string) : string;
 990: Function Get2( AURL : string) : string;
 991: Function Get8087CW : Word
 992: function GetActiveOleObject(const ClassName: String): IDispatch;
 993: Function GetAliasDriverName( const AliasName : string) : string
 994: Function GetAPMBatteryFlag : TAPMBatteryFlag
 995: Function GetAPMBatteryFullLifeTime : DWORD
 996: Function GetAPMBatteryLifePercent : Integer
 997: Function GetAPMBatteryLifeTime : DWORD
 998: Function GetAPMLineStatus : TAPMLineStatus
 999: Function GetAppdataFolder : string
1000: Function GetAppDispatcher : TComponent
1001: function GetArrayLength: integer;
1002: Function GetASCII: string;
1003: Function GetASCIILine: string;
1004: Function GetAsHandle( Format : Word) : THandle
1005: function GetAssociatedProgram(const Extension: string; var Filename, Description: string): boolean;
1006: Function GetBackupFileName( const FileName : string) : string
1007: Function GetBBitmap( Value : TBitmap) : TBitmap
1008: Function GetBIOSCopyright : string
1009: Function GetBIOSDate : TDateTime
1010: Function GetBIOSExtendedInfo : string
1011: Function GetBIOSName : string
1012: Function getBitmap(apath: string): TBitmap;
1013: Function GetBitmap( Index : Integer; Image : TBitmap) : Boolean  //object
1014: Function getBitMapObject(const bitmappath: string): TBitmap;
1015: Function GetButtonState( Button : TPageScrollerButton) : TPageScrollerButtonState
1016: Function GetCapsLockKeyState : Boolean
1017: function GetCaptureControl: TControl;
1018: Function GetCDAudioTrackList( var TrackList : TJclCdTrackInfoArray; Drive : Char) : TJclCdTrackInfo;
1019: Function GetCDAudioTrackList1( TrackList : TStrings; IncludeTrackType : Boolean; Drive : Char) : string;
1020: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char) : string
1021: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char) : string
1022: Function GetClientThread( ClientSocket : TServerClientWinSocket) : TServerClientThread
1023: Function GetClockValue : Int64
1024: function getCmdLine: PChar;
1025: function getCmdShow: Integer;
1026: function GetCPUSpeed: Double;
1027: Function GetColField( DataCol : Integer) : TField
1028: Function GetColorBlue( const Color : TColor) : Byte
1029: Function GetColorFlag( const Color : TColor) : Byte
1030: Function GetColorGreen( const Color : TColor) : Byte
1031: Function GetColorRed( const Color : TColor) : Byte
1032: Function GetComCtlVersion : Integer
1033: Function GetComPorts: TStringlist;
1034: Function GetCommonAppdataFolder : string
1035: Function GetCommonDesktopdirectoryFolder : string
1036: Function GetCommonFavoritesFolder : string
1037: Function GetCommonFilesFolder : string
1038: Function GetCommonProgramsFolder : string
1039: Function GetCommonStartmenuFolder : string
1040: Function GetCommonStartupFolder : string
1041: Function GetComponent( Owner, Parent : TComponent) : TComponent
1042: Function GetConnectionRegistryFile( DesignMode : Boolean) : string
1043: Function GetCookiesFolder : string
1044: Function GetCPUSpeed( var CpuSpeed : TFreqInfo) : Boolean
1045: Function GetCurrent : TFavoriteLinkItem
1046: Function GetCurrent : TListItem
1047: Function GetCurrent : TTaskDialogBaseButtonItem
1048: Function GetCurrent : TToolButton
1049: Function GetCurrent : TTreeNode
1050: Function GetCurrent : WideString
```

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

```
1140: Function GETIMAGELIST : TCUSTOMIMAGELIST
1141: Function GetIncome( const aNetto : Currency) : Currency
1142: Function GetIncome( const aNetto : Extended) : Extended
1143: Function GetIncome( const aNetto : Extended): Extended
1144: Function GetIncome(const aNetto : Extended) : Extended
1145: function GetIncome(const aNetto: Currency): Currency
1146: Function GetIncome2( const aNetto : Currency) : Currency
1147: Function GetIncome2( const aNetto : Currency): Currency
1148: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string) : string
1149: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN) : TINDEXDEF
1150: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet) : TIndexDef
1151: Function GetInstRes(Instance:THandle;ResType:TResType;const
      Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor):Boolean;
1152: Function
      GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
1153: Function GetIntelCacheDescription( const D : Byte) : string
1154: Function GetInteractiveUserName : string
1155: Function GetInternetCacheFolder : string
1156: Function GetInternetFormattedFileTimeStamp( const AFilename : String) : String
1157: Function GetIPAddress( const HostName : string) : string
1158: Function GetIP( const HostName : string) : string
1159: Function GetIPHostByName(const AComputerName: String): String;
1160: Function GetIsAdmin: Boolean;
1161: Function GetItem( X, Y : Integer) : LongInt
1162: Function GetItemAt( X, Y : Integer) : TListItem
1163: Function GetItemHeight(Font: TFont): Integer;
1164: Function GetItemPath( Index : Integer) : string
1165: Function GetKeyFieldNames( List : TStrings) : Integer;
1166: Function GetKeyFieldNames1( List : TWideStrings) : Integer;
1167: Function GetKeyState( const VirtualKey : Cardinal) : Boolean
1168: Function GetLastChild : LongInt
1169: Function GetLastChild : TTreeNode
1170: Function GetLastDelimitedToken( const cDelim : char; const cStr : string) : string
1171: function GetLastError: Integer
1172: Function GetLAT_CONV_FACTOR: double;  //for WGS84 power(1 - 1 / 298.257223563, 2);
1173: Function GetLoader( Ext : string) : TBitmapLoader
1174: Function GetLoadFilter : string
1175: Function GetLocalComputerName : string
1176: Function GetLocaleChar( Locale, LocaleType : Integer; Default : Char) : Char
1177: Function GetLocaleStr( Locale, LocaleType : Integer; Default : string) : string
1178: Function GetLocalUserName : string
1179: Function GetLoginUsername : WideString
1180: function getLongDayNames: string)
1181: Function GetLongHint(const hint: string): string
1182: function getLongMonthNames: string)
1183: Function GetMacAddresses( const Machine : string; const Addresses : TStrings) : Integer
1184: Function GetMainAppWndFromPid( PID : DWORD) : HWND
1185: Function GetMaskBitmap : HBITMAP
1186: Function GetMaxAppAddress : Integer
1187: Function GetMciErrorMessage( const MciErrNo : MCIERROR) : string
1188: Function GetMemoryLoad : Byte
1189: Function GetMIMEDefaultFileExt( const MIMEType : string) : TIdFileName
1190: Function GetMIMETypeFromFile( const AFile : string) : string
1191: Function GetMIMETypeFromFile( const AFile : TIdFileName) : string
1192: Function GetMinAppAddress : Integer
1193: Function GetModule : TComponent
1194: Function GetModuleHandle( ModuleName : PChar) : HMODULE
1195: Function GetModuleName( Module : HMODULE) : string
1196: Function GetModulePath( const Module : HMODULE) : string
1197: Function GetModuleFileName(Module: Integer; Filename: PChar;Size: Integer): Integer; stdcall;
1198: Function GetCommandLine: PChar; stdcall;
1199: Function GetMonochromeBitmap( Value : TBitmap) : TBitmap
1200: Function GetMultiN(aval: integer): string;
1201: Function GetName : String
1202: Function GetNearestItem( Point : TPoint; Direction : TSearchDirection) : TListItem
1203: Function GetNethoodFolder : string
1204: Function GetNext : TTreeNode
1205: Function GetNextChild( Value : LongInt) : LongInt
1206: Function GetNextChild( Value : TTreeNode) : TTreeNode
1207: Function GetNextDelimitedToken( const cDelim : char; var cStr : String) : String
1208: Function GetNextItem( StartItem: TListItem;Direction:TSearchDirection;States:TItemStates) : TListItem
1209: Function GetNextPacket : Integer
1210: Function getNextSibling : TTreeNode
1211: Function GetNextVisible : TTreeNode
1212: Function GetNode( ItemId : HTreeItem) : TTreeNode
1213: Function GetNodeAt( X, Y : Integer) : TTreeNode
1214: Function GetNodeDisplayWidth( Node : TOutlineNode) : Integer
1215: function GetNumberOfProcessors: longint;
1216: Function GetNumLockKeyState : Boolean
1217: Function GetObjectProperty( Instance : TPersistent; const PropName : string) : TObject
1218: Function GetOnlyTransitionOn( cChar : char) : TniRegularExpressionState
1219: Function GetOptionalParam( const ParamName : string) : OleVariant
1220: Function GetOSName: string;
1221: Function GetOSVersion: string;
1222: Function GetOSNumber: string;
1223: Function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
1224: Function GetPackageModuleHandle( PackageName : PChar) : HMODULE
1225: function GetPageSize: Cardinal;
1226: Function GetParameterFileName : string
```

```
1227: Function GetParams( var OwnerData : OleVariant) : OleVariant
1228: Function GETPARENTCOMPONENT : TCOMPONENT
1229: Function GetParentForm(control: TControl): TForm
1230: Function GETPARENTMENU : TMENU
1231: Function GetPassword : Boolean
1232: Function GetPassword : string
1233: Function GetPersonalFolder : string
1234: Function GetPidFromProcessName( const ProcessName : string) : DWORD
1235: Function GetPosition : TPoint
1236: Function GetPrev : TTreeNode
1237: Function GetPrevChild( Value : LongInt) : LongInt
1238: Function GetPrevChild( Value : TTreeNode) : TTreeNode
1239: Function getPrevSibling : TTreeNode
1240: Function GetPrevVisible : TTreeNode
1241: Function GetPrinthoodFolder : string
1242: Function GetPrivilegeDisplayName( const PrivilegeName : string) : string
1243: Function getProcessList: TStrings;
1244: Function GetProcessId : TIdPID
1245: Function GetProcessNameFromPid( PID : DWORD) : string
1246: Function GetProcessNameFromWnd( Wnd : HWND) : string
1247: Function GetProcessMemoryInfo(Process: THandle;ppsmemCounters: TProcessMemoryCounters;cb: DWORD):BOOL
1248: Function getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1249: Function getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1250: Function GetProgramFilesFolder : string
1251: Function GetProgramsFolder : string
1252: Function GetProxy : string
1253: Function GetQuoteChar : WideString
1254: Function GetQrCode4(Width,Height:Word; Correct_Level:string;const
      Data:string;aformat:string):TLinearBitmap;
1255: Function GetQrCodetoFile(Width,Height:Word;Correct_Level:string;const
      Data:string;aformat:string):TLinearBitmap;
1256: Function GetRate : Double
1257: Function getPerfTime: string;
1258: Function getRuntime: string;
1259: Function GetRBitmap( Value : TBitmap) : TBitmap
1260: Function GetReadableName( const AName : string) : string
1261: Function GetRecentDocs : TStringList
1262: Function GetRecentFolder : string
1263: Function GetRecords( Count : Integer; out RecsOut : Integer; Options : Integer) : OleVariant;
1264: Function GetRecords1(Count:Integer; out RecsOut:Integer;Options:Integer;const CommandText:WideString;var
      Params, OwnerData : OleVariant) : OleVariant;
1265: Function GetRecordset( const CommandText : WideString; ConnectionString : WideString) : _Recordset
1266: Function GetRegisteredCompany : string
1267: Function GetRegisteredOwner : string
1268: Function GetResource(ResType:TResType;const
      Name:string;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor:Bool
1269: Function GetResourceName( ObjStream : TStream; var AName : string) : Boolean
1270: Function GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
1271: Function GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
1272: Function GetRValue( rgb : DWORD) : Byte
1273: Function GetGValue( rgb : DWORD) : Byte
1274: Function GetBValue( rgb : DWORD) : Byte
1275: Function GetCValue( cmyk : COLORREF) : Byte
1276: Function GetMValue( cmyk : COLORREF) : Byte
1277: Function GetYValue( cmyk : COLORREF) : Byte
1278: Function GetKValue( cmyk : COLORREF) : Byte
1279: Function CMYK( c, m, y, k : Byte) : COLORREF
1280: Function GetOSName: string;
1281: Function GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR) : FARPROC
1282: Function GetProcAddress(Module : HMODULE; Proc : PChar): Dword
1283: Function GetSafeCallExceptionMsg : String
1284: Function GetSaturationBitmap( Value : TBitmap) : TBitmap
1285: Function GetSaveFilter : string
1286: Function GetSaver( Ext : string) : TBitmapLoader
1287: Function GetScrollLockKeyState : Boolean
1288: Function GetSearchString : string
1289: Function GetSelections( AList : TList) : TTreeNode
1290: function GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1291: Function GetSendToFolder : string
1292: Function GetServer : IAppServer
1293: Function GetServerList : OleVariant
1294: Function GetShadowColor( const Color : TColor; Luminance : Integer) : TColor
1295: Function GetShellProcessHandle : THandle
1296: Function GetShellProcessName : string
1297: Function GetShellVersion : Cardinal
1298: function getShortDayNames: string)
1299: Function GetShortHint(const hint: string): string
1300: function getShortMonthNames: string)
1301: Function GetSizeOfFile( const FileName : string) : Int64;
1302: Function GetSizeOfFile1( Handle : THandle) : Int64;
1303: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1304: Function GetStartmenuFolder : string
1305: Function GetStartupFolder : string
1306: Function GetStringProperty( Instance : TPersistent; const PropName : string) : WideString
1307: Function GetSuccessor( cChar : char) : TniRegularExpressionState
1308: Function GetSwapFileSize : Integer
1309: Function GetSwapFileUsage : Integer
1310: Function GetSystemLocale : TIdCharSet
1311: Function GetSystemMetrics( nIndex : Integer) : Integer
```

```
1312: Function GetSystemPathSH(Folder: Integer): TFilename ;
1313: Function GetTableNameFromQuery( const SQL : Widestring) : Widestring
1314: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1315: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1316: Function GetTasksList( const List : TStrings) : Boolean
1317: Function getTeamViewerID: string;
1318: Function GetTemplatesFolder : string
1319: Function GetText : PwideChar
1320: function GetText:PChar
1321: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1322: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1323: Function GetTextItem( const Value : string) : Longint
1324: function GETTEXTLEN:INTEGER
1325: Function GetThreadLocale: Longint; stdcall
1326: Function GetCurrentThreadID: LongWord; stdcall;
1327: Function GetTickCount : Cardinal
1328: Function GetTickDiff( const AOldTickCount, ANewTickCount : Cardinal) : Cardinal
1329: Function GetTicketNr : longint
1330: Function GetTime : Cardinal
1331: Function GetTime : TDateTime
1332: Function GetTimeout : Integer
1333: Function GetTimeStr: String
1334: Function GetTimeString: String
1335: Function GetTodayFiles(startdir, amask: string): TStringlist;
1336: Function getTokenCounts : integer
1337: Function GetTotalPageFileMemory : Integer
1338: Function GetTotalPhysicalMemory : Integer
1339: Function GetTotalVirtualMemory : Integer
1340: Function GetUniqueFileName( const APath, APrefix, AExt : String) : String
1341: Function GetUseNowForDate : Boolean
1342: Function GetUserDomainName( const CurUser : string) : string
1343: Function GetUserName : string
1344: Function GetUserName: string;
1345: Function GetUserObjectName( hUserObject : THandle) : string
1346: Function GetValueBitmap( Value : TBitmap) : TBitmap
1347: Function GetValueMSec : Cardinal
1348: Function GetValueStr : String
1349: Function GetVersion: int;
1350: Function GetVersionString(FileName: string): string;
1351: Function getVideoDrivers: string;
1352: Function GetVisibleNode( Index : LongInt) : TOutlineNode
1353: Function GetVolumeFileSystem( const Drive : string) : string
1354: Function GetVolumeName( const Drive : string) : string
1355: Function GetVolumeSerialNumber( const Drive : string) : string
1356: Function GetWebAppServices : IWebAppServices
1357: Function GetWebRequestHandler : IWebRequestHandler
1358: Function GetWindowCaption( Wnd : HWND) : string
1359: Function GetWindowDC(hdwnd: HWND): HDC;
1360: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1361: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1362: Function GetWindowsComputerID : string
1363: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1364: Function GetWindowsFolder : string
1365: Function GetWindowsServicePackVersion : Integer
1366: Function GetWindowsServicePackVersionString : string
1367: Function GetWindowsSystemFolder : string
1368: Function GetWindowsTempFolder : string
1369: Function GetWindowsUserID : string
1370: Function GetWindowsVersion : TWindowsVersion
1371: Function GetWindowsVersionString : string
1372: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1373: Function GMTToLocalDateTime( S : string) : TDateTime
1374: Function GotoKey : Boolean
1375: Function GradToCycle( const Grads : Extended) : Extended
1376: Function GradToDeg( const Grads : Extended) : Extended
1377: Function GradToDeg( const Value : Extended) : Extended;
1378: Function GradToDeg1( const Value : Double) : Double;
1379: Function GradToDeg2( const Value : Single) : Single;
1380: Function GradToRad( const Grads : Extended) : Extended
1381: Function GradToRad( const Value : Extended) : Extended;
1382: Function GradToRad1( const Value : Double) : Double;
1383: Function GradToRad2( const Value : Single) : Single;
1384: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1385: Function GreenComponent( const Color32 : TColor32) : Integer
1386: function GUIDToString(const GUID: TGUID): string)
1387: Function HandleAllocated : Boolean
1388: function HandleAllocated: Boolean;
1389: Function HandleRequest : Boolean
1390: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1391: Function HarmonicMean( const X : TDynFloatArray) : Float
1392: Function HasAsParent( Value : TTreeNode) : Boolean
1393: Function HASCHILDDEFS : BOOLEAN
1394: Function HasCurValues : Boolean
1395: Function HasExtendCharacter( const s : UTF8String) : Boolean
1396: Function HasFormat( Format : Word) : Boolean
1397: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1398: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1399: Function HashValue(AStream: TStream): LongWord
1400: Function HashValue(AStream: TStream): Word
```

```
1401: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1402: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1403: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1404: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1405: Function HashValue16( const ASrc : string) : Word;
1406: Function HashValue16stream( AStream : TStream) : Word;
1407: Function HashValue32( const ASrc : string) : LongWord;
1408: Function HashValue32Stream( AStream : TStream) : LongWord;
1409: Function HasMergeConflicts : Boolean
1410: Function hasMoreTokens : boolean
1411: Function HASPARENT : BOOLEAN
1412: function HasParent: Boolean
1413: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1414: Function HasUTF8BOM( S : TStream) : boolean;
1415: Function HasUTF8BOM1( S : AnsiString) : boolean;
1416: Function Haversine( X : Float) : Float
1417: Function Head( s : string; const subs : string; var tail : string) : string
1418: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1419: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1420: function HELPJUMP(JUMPID:STRING):BOOLEAN
1421: Function HeronianMean( const a, b : Float) : Float
1422: function HexStrToStr(Value: string): string;
1423: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1424: function HexToBin2(HexNum: string): string;
1425: Function HexToDouble( const Hex : string) : Double
1426: function HexToInt(hexnum: string): LongInt;
1427: function HexToStr(Value: string): string;
1428: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1429: function Hi(vdat: word): byte;
1430: function HiByte(W: Word): Byte)
1431: function High: Int64;
1432: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1433: function HINSTANCE: longword;
1434: function HiWord(l: DWORD): Word)
1435: function HMODULE: longword;
1436: Function HourOf( const AValue : TDateTime) : Word
1437: Function HourOfTheDay( const AValue : TDateTime) : Word
1438: Function HourOfTheMonth( const AValue : TDateTime) : Word
1439: Function HourOfTheWeek( const AValue : TDateTime) : Word
1440: Function HourOfTheYear( const AValue : TDateTime) : Word
1441: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1442: Function HourSpan( const ANow, AThen : TDateTime) : Double
1443: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1444: Function HTMLDecode( const AStr : String) : String
1445: Function HTMLEncode( const AStr : String) : String
1446: Function HTMLEscape( const Str : string) : string
1447: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1448: Function HTTPDecode( const AStr : String) : string
1449: Function HTTPEncode( const AStr : String) : string
1450: Function Hypot( const X, Y : Extended) : Extended
1451: Function IBMax( n1, n2 : Integer) : Integer
1452: Function IBMin( n1, n2 : Integer) : Integer
1453: Function IBRandomString( iLength : Integer) : String
1454: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1455: Function IBStripString( st : String; CharsToStrip : String) : String
1456: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1457: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1458: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1459: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1460: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1461: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1462:  Function RandomString( iLength : Integer) : String');
1463:  Function RandomInteger( iLow, iHigh : Integer) : Integer');
1464:  Function StripString( st : String; CharsToStrip : String) : String');
1465:  Function FormatIdentifier( Dialect : Integer; Value : String) : String');
1466:  Function FormatIdentifierValue( Dialect : Integer; Value : String) : String');
1467:  Function ExtractIdentifier( Dialect : Integer; Value : String) : String');
1468:  Function QuoteIdentifier( Dialect : Integer; Value : String) : String');
1469:  Function AddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string;
1470:  Procedure DecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String;
1471:  function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;
1472: Function IconToBitmap( Ico : HICON) : TBitmap
1473: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1474: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1475: function IdentToCharset(const Ident: string; var Charset: Longint): Boolean)
1476: function IdentToColor(const Ident: string; var Color: Longint): Boolean)
1477: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1478: Function IdGetDefaultCharSet : TIdCharSet
1479: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1480: Function IdPorts2 : TStringList
1481: Function IdToMib( const Id : string) : string
1482: Function IdSHA1Hash(apath: string): string;
1483: Function IdHashSHA1(apath: string): string;
1484: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1485: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1486: Function IfThenInt( AValue : Boolean; const ATrue : integer; AFalse : integer): integer;');
1487: Function IfThenDouble( AValue : Boolean; const ATrue : double; AFalse : double): double;');
1488: Function IfThenBool( AValue : Boolean; const ATrue : boolean; AFalse : boolean): boolean;');
1489: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer;
```

```
1490: Function iif2( ATest : Boolean; const ATrue : string; const AFalse : string) : string;
1491: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFalse : Boolean) : Boolean;
1492: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;
1493: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer) : TDateTime
1494: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1495: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1496: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1497: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1498: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1499: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1500: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1501: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1502: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1503: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1504: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1505: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1506: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1507: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1508: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1509: Function IncludeTrailingBackslash( S : string) : string
1510: function IncludeTrailingBackslash(const S: string): string)
1511: Function IncludeTrailingPathDelimiter( const APath : string) : string
1512: Function IncludeTrailingPathDelimiter( S : string) : string
1513: function IncludeTrailingPathDelimiter(const S: string): string)
1514: Function IncludeTrailingSlash( const APath : string) : string
1515: Function IncMilliSecond( const AValue : TDateTime; const ANumberOfMilliSeconds : Int64) : TDateTime
1516: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64) : TDateTime
1517: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer) : TDateTime
1518: function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime)
1519: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64) : TDateTime
1520: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer) : TDateTime
1521: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer) : TDateTime
1522: Function IndexOf( AClass : TClass) : Integer
1523: Function IndexOf( AComponent : TComponent) : Integer
1524: Function IndexOf( AObject : TObject) : Integer
1525: Function INDEXOF( const ANAME : String) : INTEGER
1526: Function IndexOf( const DisplayName : string) : Integer
1527: Function IndexOf( const Item : TBookmarkStr) : Integer
1528: Function IndexOf( const S : WideString) : Integer
1529: Function IndexOf( const View : TJclFileMappingView) : Integer
1530: Function INDEXOF( FIELD : TFIELD) : INTEGER
1531: Function IndexOf( ID : LCID) : Integer
1532: Function INDEXOF( ITEM : TMENUITEM) : INTEGER
1533: Function IndexOf( Value : TListItem) : Integer
1534: Function IndexOf( Value : TTreeNode) : Integer
1535: function IndexOf(const S: string): Integer;
1536: Function IndexOfName( const Name : WideString) : Integer
1537: function IndexOfName(Name: string): Integer;
1538: Function IndexOfObject( AObject : TObject) : Integer
1539: function IndexofObject(AObject:tObject):Integer
1540: Function IndexOfTabAt( X, Y : Integer) : Integer
1541: Function IndexStr( const AText : string; const AValues : array of string) : Integer
1542: Function IndexText( const AText : string; const AValues : array of string) : Integer
1543: Function IndexOfInteger( AList : TStringList; Value : Variant) : Integer
1544: Function IndexOfFloat( AList : TStringList; Value : Variant) : Integer
1545: Function IndexOfDate( AList : TStringList; Value : Variant) : Integer
1546: Function IndexOfString( AList : TStringList; Value : Variant) : Integer
1547: Function IndyCompareStr( const A1 : string; const A2 : string) : Integer
1548: Function IndyGetHostName : string
1549: Function IndyInterlockedDecrement( var I : Integer) : Integer
1550: Function IndyInterlockedExchange( var A : Integer; B : Integer) : Integer
1551: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer) : Integer
1552: Function IndyInterlockedIncrement( var I : Integer) : Integer
1553: Function IndyLowerCase( const A1 : string) : string
1554: Function IndyStrToBool( const AString : String) : Boolean
1555: Function IndyUpperCase( const A1 : string) : string
1556: Function InitCommonControl( CC : Integer) : Boolean
1557: Function InitTempPath : string
1558: Function InMainThread : boolean
1559: Function inOpArray( W : WideChar; sets : array of WideChar) : boolean
1560: Function Input: Text)
1561: Function InputBox( const ACaption, APrompt, ADefault : string) : string
1562: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1563: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1564: Function InputQuery( const ACaption, APrompt : string; var Value : string) : Boolean
1565: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1566: Function InquireSignal( RtlSigNum : Integer) : TSignalState
1567: Function InRangeR( const A, Min, Max : Double) : Boolean
1568: function Insert( Index : Integer) : TCollectionItem
1569: Function Insert( Index : Integer) : TComboExItem
1570: Function Insert( Index : Integer) : THeaderSection
1571: Function Insert( Index : Integer) : TListItem
1572: Function Insert( Index : Integer) : TStatusPanel
1573: Function Insert( Index : Integer) : TWorkArea
1574: Function Insert( Index : LongInt; const Text : string) : LongInt
1575: Function Insert( Sibling : TTreeNode; const S : string) : TTreeNode
1576: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM) : INTEGER
1577: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : INTEGER
1578: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
```

```
1579: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
1580: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1581: Function Instance : Longint
1582: function InstanceSize: Longint
1583: Function Int(e : Extended) : Extended;
1584: function Int64ToStr(i: Int64): String;
1585: Function IntegerToBcd( const AValue : Integer) : TBcd
1586: Function Intensity( const Color32 : TColor32) : Integer;
1587: Function Intensity( const R, G, B : Single) : Single;
1588: Function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
      FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1589: Function InterestRate(NPeriods:Integer;const Payment,PresVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
1590: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1591: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1592: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1593: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1594: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1595: Function IntMibToStr( const Value : string) : string
1596: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1597: Function IntToBin( Value : cardinal) : string
1598: Function IntToHex( Value : Integer; Digits : Integer) : string;
1599: function IntToHex(a: integer; b: integer): string;
1600: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1601: function IntToHex64(Value: Int64; Digits: Integer): string)
1602: Function IntTo3Str( Value : Longint; separator: string) : string
1603: Function inttobool( aInt : LongInt) : Boolean
1604: function IntToStr(i: Int64): String;
1605: Function IntToStr64(Value: Int64): string)
1606: function IOResult: Integer
1607: Function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1608: Function IsAccel(VK: Word; const Str: string): Boolean
1609: Function IsAddressInNetwork( Address : String) : Boolean
1610: Function IsAdministrator : Boolean
1611: Function IsAlias( const Name : string) : Boolean
1612: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1613: Function IsASCII( const AByte : Byte) : Boolean;
1614: Function IsASCIILDH( const AByte : Byte) : Boolean;
1615: Function IsAssembly(const FileName: string): Boolean;
1616: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1617: Function IsBinary(const AChar : Char) : Boolean
1618: function IsConsole: Boolean)
1619: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1620: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1621: Function IsDelphiDesignMode : boolean
1622: Function IsDelphiRunning : boolean
1623: Function IsDFAState : boolean
1624: Function IsDirectory( const FileName : string) : Boolean
1625: Function IsDomain( const S : String) : Boolean
1626: function IsDragObject(Sender: TObject): Boolean;
1627: Function IsEditing : Boolean
1628: Function ISEMPTY : BOOLEAN
1629: Function IsEqual( Value : TParameters) : Boolean
1630: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1631: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1632: Function IsFirstNode : Boolean
1633: Function IsFloatZero( const X : Float) : Boolean
1634: Function IsFormatRegistered( Extension, AppID : string) : Boolean
1635: Function IsFormOpen(const FormName: string): Boolean;
1636: Function IsFQDN( const S : String) : Boolean
1637: Function IsGrayScale : Boolean
1638: Function IsHex( AChar : Char) : Boolean;
1639: Function IsHexString(const AString: string): Boolean;
1640: Function IsHostname( const S : String) : Boolean
1641: Function IsInfinite( const AValue : Double) : Boolean
1642: Function IsInLeapYear( const AValue : TDateTime) : Boolean
1643: Function IsInternet: boolean;
1644: Function IsLeadChar( ACh : Char) : Boolean
1645: Function IsLeapYear( Year : Word) : Boolean
1646: function IsLeapYear(Year: Word): Boolean)
1647: function IsLibrary: Boolean)
1648: Function ISLINE : BOOLEAN
1649: Function IsLinkedTo( DataSet : TDataSet) : Boolean
1650: Function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1651: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1652: Function IsMatch( const Pattern, Text : string) : Boolean  //Grep like RegEx
1653: Function IsMainAppWindow( Wnd : HWND) : Boolean
1654: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1655: function IsMemoryManagerSet: Boolean)
1656: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1657: function IsMultiThread: Boolean)
1658: Function IsNumeric( AChar : Char) : Boolean;
1659: Function IsNumeric2( const AString : string) : Boolean;
1660: Function IsNTFS: Boolean;
1661: Function IsOctal( AChar : Char) : Boolean;
1662: Function IsOctalString(const AString: string) : Boolean;
1663: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1664: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1665: Function IsPM( const AValue : TDateTime) : Boolean
```

```
1666: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1667: Function IsPortAvailable( ComNum : Cardinal) : Boolean');
1668: Function IsCOMPortReal( ComNum : Cardinal) : Boolean');
1669: Function IsCOM( ComNum : Cardinal) : Boolean');
1670: Function IsCOMPort: Boolean');
1671: Function IsPrimeFactor( const F, N : Cardinal) : Boolean
1672: Function IsPrimeRM( N : Cardinal) : Boolean  //rabin miller
1673: Function IsPrimeTD( N : Cardinal) : Boolean  //trial division
1674: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1675: Function ISqrt( const I : Smallint) : Smallint
1676: Function IsReadOnly(const Filename: string): boolean;
1677: Function IsRectEmpty( const Rect : TRect) : Boolean
1678: function IsRectEmpty(const Rect: TRect): Boolean)
1679: Function IsRelativePrime( const X, Y : Cardinal) : Boolean
1680: Function ISRIGHTTOLEFT : BOOLEAN
1681: function IsRightToLeft: Boolean
1682: Function IsSameDay( const AValue, ABasis : TDateTime) : Boolean
1683: Function ISSEQUENCED : BOOLEAN
1684: Function IsSystemModule( const Module : HMODULE) : Boolean
1685: Function IsSystemResourcesMeterPresent : Boolean
1686: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1687: Function IsToday( const AValue : TDateTime) : Boolean
1688: function IsToday(const AValue: TDateTime): Boolean;
1689: Function IsTopDomain( const AStr : string) : Boolean
1690: Function IsUTF8LeadByte( Lead : Char) : Boolean
1691: Function IsUTF8String( const s : UTF8String) : Boolean
1692: Function IsUTF8TrailByte( Lead : Char) : Boolean
1693: Function ISVALIDCHAR( INPUTCHAR : CHAR) : BOOLEAN
1694: Function IsValidDate( const AYear, AMonth, ADay : Word) : Boolean
1695: Function IsValidDateDay( const AYear, ADayOfYear : Word) : Boolean
1696: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : Boolean
1697: Function IsValidDateTime(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1698: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) : Boolean
1699: Function IsValidIdent( Ident : string) : Boolean
1700: function IsValidIdent(const Ident: string; AllowDots: Boolean): Boolean)
1701: Function IsValidIP( const S : String) : Boolean
1702: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) : Boolean
1703: Function IsValidISBN( const ISBN : AnsiString) : Boolean
1704: Function IsVariantManagerSet: Boolean; //deprecated;
1705: Function IsVirtualPcGuest : Boolean;
1706: Function IsVmWareGuest : Boolean;
1707: Function IsVCLControl(Handle: HWnd): Boolean;
1708: Function IsWhiteString( const AStr : String) : Boolean
1709: Function IsWindowResponding( Wnd : HWND; Timeout : Integer) : Boolean
1710: Function IsWoW64: boolean;
1711: Function IsWin64: boolean;
1712: Function IsWow64String(var s: string): Boolean;
1713: Function IsWin64String(var s: string): Boolean;
1714: Function IsWindowsVista: boolean;
1715: Function isPowerof2(num: int64): boolean;
1716: Function powerOf2(exponent: integer): int64;
1717: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1718: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1719: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1720: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1721: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1722: Function ItemRect( Index : Integer) : TRect
1723: function ITEMRECT(INDEX:INTEGER):TRECT
1724: Function ItemWidth( Index : Integer) : Integer
1725: Function JavahashCode(val: string): Integer;
1726: Function JosephusG(n,k: integer; var graphout: string): integer;
1727: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1728: Function JustName(PathName : string) : string;  //in path and ext
1729: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1730: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1731: Function KeepAlive : Boolean
1732: Function KeysToShiftState(Keys: Word): TShiftState;
1733: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1734: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1735: Function KeyboardStateToShiftState: TShiftState; overload;
1736: Function Languages : TLanguages
1737: Function Last : TClass
1738: Function Last : TComponent
1739: Function Last : TObject
1740: Function LastDelimiter( Delimiters, S : string) : Integer
1741: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1742: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1743: Function Latitude2WGS84(lat: double): double;
1744: Function LCM(m,n:longint):longint;
1745: Function LCMJ( const X, Y : Cardinal) : Cardinal
1746: Function Ldexp( const X : Extended; const P : Integer) : Extended
1747: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1748: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1749: function Length: Integer;
1750: Procedure LetFileList(FileList: TStringlist; apath: string);
1751: function lengthmp3(mp3path: string):integer;
1752: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1753: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1754: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
```

```
1755: function LineStart(Buffer, BufPos: PChar): PChar
1756: function LineStart(Buffer, BufPos: PChar): PChar)
1757: function ListSeparator: char;
1758: function Ln(x: Extended): Extended;
1759: Function LnXP1( const X : Extended) : Extended
1760: function Lo(vdat: word): byte;
1761: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1762: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1763: Function LoadFileAsString( const FileName : string) : string
1764: Function LoadFromFile( const FileName : string) : TBitmapLoader
1765: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1766: Function LoadPackage(const Name: string): HMODULE
1767: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1768: Function LoadStr( Ident : Integer) : string
1769: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1770: Function LoadWideStr( Ident : Integer) : WideString
1771: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1772: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1773: Function LockServer( fLock : LongBool) : HResult
1774: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1775: Function Log( const X : Extended) : Extended
1776: Function Log10( const X : Extended) : Extended
1777: Function Log2( const X : Extended) : Extended
1778: function LogBase10(X: Float): Float;
1779: Function LogBase2(X: Float): Float;
1780: Function LogBaseN(Base, X: Float): Float;
1781: Function LogN( const Base, X : Extended) : Extended
1782: Function LogOffOS : Boolean
1783: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1784: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1785: Function LongDateFormat: string;
1786: function LongTimeFormat: string;
1787: Function LongWordToFourChar( ACardinal : LongWord) : string
1788: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1789: Function LookupName( const name : string) : TInAddr
1790: Function LookupService( const service : string) : Integer
1791: function Low: Int64;
1792: Function LowerCase( S : string) : string
1793: Function Lowercase(s : AnyString) : AnyString;
1794: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1795: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1796: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1797: function MainInstance: longword
1798: function MainThreadID: longword
1799: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1800: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1801: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1802: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1803: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1804: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1805: Function MakeFile(const FileName: string): integer)');
1806: function MakeLong(A, B: Word): Longint)
1807: Function MakeTempFilename( const APath : String) : string
1808: Function MakeValidFileName( const Str : string) : string
1809: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1810: function MakeWord(A, B: Byte): Word)
1811: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1812: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1813: Function MapValues( Mapping : string; Value : string) : string
1814: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1815: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1816: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1817: Function MaskGetFldSeparator( const EditMask : string) : Integer
1818: Function MaskGetMaskBlank( const EditMask : string) : Char
1819: Function MaskGetMaskSave( const EditMask : string) : Boolean
1820: Function MaskIntlLiteralToChar( IChar : Char) : Char
1821: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1822: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1823: Function MaskString( Mask, Value : String) : String
1824: Function Match( const sString : string) : TniRegularExpressionMatchResul
1825: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1826: Function Matches( const Filename : string) : Boolean
1827: Function MatchesMask( const Filename, Mask : string) : Boolean
1828: Function MatchStr( const AText : string; const AValues : array of string) : Boolean
1829: Function MatchText( const AText : string; const AValues : array of string) : Boolean
1830: Function Max( AValueOne, AValueTwo : Integer) : Integer
1831: function Max(const x,y: Integer): Integer;
1832: Function Max1( const B1, B2 : Shortint) : Shortint;
1833: Function Max2( const B1, B2 : Smallint) : Smallint;
1834: Function Max3( const B1, B2 : Word) : Word;
1835: function Max3(const x,y,z: Integer): Integer;
1836: Function Max4( const B1, B2 : Integer) : Integer;
1837: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1838: Function Max6( const B1, B2 : Int64) : Int64;
1839: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1840: Function MaxFloat( const X, Y : Float) : Float
1841: Function MaxFloatArray( const B : TDynFloatArray) : Float
1842: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1843: function MaxIntValue(const Data: array of Integer):Integer)
```

```
1844: Function MaxJ( const B1, B2 : Byte) : Byte;
1845: function MaxPath: string;
1846: function MaxValue(const Data: array of Double): Double)
1847: Function MaxCalc( const Formula : string) : Extended  //math expression parser
1848: Procedure MaxCalcF( const Formula : string);   //out to console memo2
1849: function MD5(const fileName: string): string;
1850: Function Mean( const Data : array of Double) : Extended
1851: Function Median( const X : TDynFloatArray) : Float
1852: Function Memory : Pointer
1853: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1854: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1855: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1856: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1857: Function MessageDlg1(const
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1858: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1859: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
1860: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1861: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1862: Function MibToId( Mib : string) : string
1863: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1864: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1865: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1866: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1867: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1868: Procedure GetMidiOutputs( const List : TStrings)
1869: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1870: Function MIDINoteToStr( Note : TMIDINote) : string
1871: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1872: Procedure GetMidiOutputs( const List : TStrings)
1873: Procedure MidiOutCheck( Code : MMResult)
1874: Procedure MidiInCheck( Code : MMResult)
1875: Function MilliSecondOf( const AValue : TDateTime) : Word
1876: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1877: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1878: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1879: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1880: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
1881: Function MilliSecondOfTheWeek( const AValue : TDateTime) : LongWord
1882: Function MilliSecondOfTheYear( const AValue : TDateTime) : Int64
1883: Function MilliSecondsBetween( const ANow, AThen : TDateTime) : Int64
1884: Function MilliSecondSpan( const ANow, AThen : TDateTime) : Double
1885: Function milliToDateTime( MilliSecond : LongInt) : TDateTime');
1886: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
1887: Function millis: int64;
1888: Function Min( AValueOne, AValueTwo : Integer) : Integer
1889: Function Min1( const B1, B2 : Shortint) : Shortint;
1890: Function Min2( const B1, B2 : Smallint) : Smallint;
1891: Function Min3( const B1, B2 : Word) : Word;
1892: Function Min4( const B1, B2 : Integer) : Integer;
1893: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1894: Function Min6( const B1, B2 : Int64) : Int64;
1895: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1896: Function MinClientRect : TRect;
1897: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1898: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1899: Function MinFloat( const X, Y : Float) : Float
1900: Function MinFloatArray( const B : TDynFloatArray) : Float
1901: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1902: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
1903: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1904: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1905: Function MinIntValue( const Data : array of Integer) : Integer
1906: function MinIntValue(const Data: array of Integer):Integer)
1907: Function MinJ( const B1, B2 : Byte) : Byte;
1908: Function MinuteOf( const AValue : TDateTime) : Word
1909: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1910: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1911: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1912: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1913: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1914: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1915: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1916: Function MinValue( const Data : array of Double) : Double
1917: function MinValue(const Data: array of Double): Double)
1918: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1919: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1920: Function ModFloat( const X, Y : Float) : Float
1921: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1922: Function Modify( const Key : string; Value : Integer) : Boolean
1923: Function ModuleCacheID : Cardinal
1924: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1925: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1926: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1927: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
```

```
1928: Function MonthOf( const AValue : TDateTime) : Word
1929: Function MonthOfTheYear( const AValue : TDateTime) : Word
1930: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1931: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1932: Function MonthStr( DateTime : TDateTime) : string
1933: Function MouseCoord( X, Y : Integer) : TGridCoord
1934: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1935: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1936: Function MoveNext : Boolean
1937: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1938: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1939: Function Name : string
1940: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
      Double;PaymentTime:TPaymentTime):Extended
1941: function NetworkVolume(DriveChar: Char): string
1942: Function NEWBOTTOMLINE : INTEGER
1943: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1944: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
      TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1945: Function NEWLINE : TMENUITEM
1946: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1947: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
      Right:PExprNode):PExprNode
1948: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
      const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1949: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
1950: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
      TMenuItem;AENABLED:BOOL): TMENUITEM
1951: Function NEWTOPLINE : INTEGER
1952: Function Next : TIdAuthWhatsNext
1953: Function NextCharIndex( S : String; Index : Integer) : Integer
1954: Function NextRecordSet : TCustomSQLDataSet
1955: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1956: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
1957: Function NextToken : Char
1958: Function nextToken : WideString
1959: function NextToken:Char
1960: Function Norm( const Data : array of Double) : Extended
1961: Function NormalizeAngle( const Angle : Extended) : Extended
1962: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
1963: Function NormalizeRect( const Rect : TRect) : TRect
1964: function NormalizeRect(const Rect: TRect): TRect;
1965: Function Now : TDateTime
1966: function Now2: tDateTime
1967: Function NumProcessThreads : integer
1968: Function NumThreadCount : integer
1969: Function NthDayOfWeek( const AValue : TDateTime) : Word
1970: Function NtProductType : TNtProductType
1971: Function NtProductTypeString : string
1972: function Null: Variant;
1973: Function NullPoint : TPoint
1974: Function NullRect : TRect
1975: Function Null2Blank(aString:String):String;
1976: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
1977: Function NumIP : integer
1978: function Odd(x: Longint): boolean;
1979: Function OffsetFromUTC : TDateTime
1980: Function OffsetPoint( const P, Offset : TPoint) : TPoint
1981: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
1982: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
1983: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
1984: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
1985: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
1986: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
1987: function OpenBit:Integer
1988: Function OpenDatabase : TDatabase
1989: Function OpenDatabase( const DatabaseName : string) : TDatabase
1990: Procedure OpenDir(adir: string);
1991: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
1992: Function OpenObject( Value : PChar) : Boolean;
1993: Function OpenObject1( Value : string) : Boolean;
1994: Function OpenSession( const SessionName : string) : TSession
1995: Function OpenVolume( const Drive : Char) : THandle
1996: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
1997: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
1998: Function OrdToBinary( const Value : Byte) : string;
1999: Function OrdToBinary1( const Value : Shortint) : string;
2000: Function OrdToBinary2( const Value : Smallint) : string;
2001: Function OrdToBinary3( const Value : Word) : string;
2002: Function OrdToBinary4( const Value : Integer) : string;
2003: Function OrdToBinary5( const Value : Cardinal) : string;
2004: Function OrdToBinary6( const Value : Int64) : string;
2005: Function OSCheck( RetVal : Boolean) : Boolean
2006: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
2007: Function OSIdentToString( const OSIdent : DWORD) : string
2008: Function Output: Text)
2009: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
2010: Function Owner : TCustomListView
```

```
2011: function Owner : TPersistent
2012: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
2013: Function PadL( pStr : String; pLth : integer) : String
2014: Function Padl(s : AnyString;I : longInt) : AnyString;
2015: Function PadLCh( pStr : String; pLth : integer; pChr : char) : String
2016: Function PadR( pStr : String; pLth : integer) : String
2017: Function Padr(s : AnyString;I : longInt) : AnyString;
2018: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
2019: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2020: Function Padz(s : AnyString;I : longInt) : AnyString;
2021: Function PaethPredictor( a, b, c : Byte) : Byte
2022: Function PARAMBYNAME( const VALUE : String) : TPARAM
2023: Function ParamByName( const Value : WideString) : TParameter
2024: Function ParamCount: Integer
2025: Function ParamsEncode( const ASrc : string) : string
2026: function ParamStr(Index: Integer): string)
2027: Function ParseDate( const DateStr : string) : TDateTime
2028: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
2029: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2030: Function PathAddExtension( const Path, Extension : string) : string
2031: Function PathAddSeparator( const Path : string) : string
2032: Function PathAppend( const Path, Append : string) : string
2033: Function PathBuildRoot( const Drive : Byte) : string
2034: Function PathCanonicalize( const Path : string) : string
2035: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2036: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2037: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2038: Function PathEncode( const ASrc : string) : string
2039: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2040: Function PathExtractFileNameNoExt( const Path : string) : string
2041: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2042: Function PathGetDepth( const Path : string) : Integer
2043: Function PathGetLongName( const Path : string) : string
2044: Function PathGetLongName2( Path : string) : string
2045: Function PathGetShortName( const Path : string) : string
2046: Function PathIsAbsolute( const Path : string) : Boolean
2047: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2048: Function PathIsDiskDevice( const Path : string) : Boolean
2049: Function PathIsUNC( const Path : string) : Boolean
2050: Function PathRemoveExtension( const Path : string) : string
2051: Function PathRemoveSeparator( const Path : string) : string
2052: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2053: Function Peek : Pointer
2054: Function Peek : TObject
2055: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2056: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2057: function Permutation(npr, k: integer): extended;
2058: function PermutationInt(npr, k: integer): Int64;
2059: Function PermutationJ( N, R : Cardinal) : Float
2060: Function Pi : Extended;
2061: Function PiE : Extended;
2062: Function PixelsToDialogsX( const Pixels : Word) : Word
2063: Function PixelsToDialogsY( const Pixels : Word) : Word
2064: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2065: Function Point( X, Y : Integer) : TPoint
2066: function Point(X, Y: Integer): TPoint)
2067: Function PointAssign( const X, Y : Integer) : TPoint
2068: Function PointDist( const P1, P2 : TPoint) : Double;
2069: function PointDist(const P1,P2: TFloatPoint): Double;
2070: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2071: function PointDist2(const P1,P2: TPoint): Double;
2072: Function PointEqual( const P1, P2 : TPoint) : Boolean
2073: Function PointIsNull( const P : TPoint) : Boolean
2074: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2075: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2076: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2077: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2078: Function Pop : Pointer
2079: Function Pop : TObject
2080: Function PopnStdDev( const Data : array of Double) : Extended
2081: Function PopnVariance( const Data : array of Double) : Extended
2082: Function PopulationVariance( const X : TDynFloatArray) : Float
2083: function Pos(SubStr, S: AnyString): Longint;
2084: Function PosEqual( const Rect : TRect) : Boolean
2085: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2086: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2087: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2088: Function Post1( AURL : string; const ASource : TStrings) : string;
2089: Function Post2( AURL : string; const ASource : TStream) : string;
2090: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream) : string;
2091: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2092: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2093: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2094: Function Power( const Base, Exponent : Extended) : Extended
2095: Function PowerBig(aval, n:integer): string;
2096: Function PowerIntJ( const X : Float; N : Integer) : Float;
2097: Function PowerJ( const Base, Exponent : Float) : Float;
```

```
2098: Function PowerOffOS : Boolean
2099: Function PreformatDateString( Ps : string) : string
2100: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2101: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2102: Function Printer : TPrinter
2103: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2104: Function ProcessResponse : TIdHTTPWhatsNext
2105: Function ProduceContent : string
2106: Function ProduceContentFromStream( Stream : TStream) : string
2107: Function ProduceContentFromString( const S : string) : string
2108: Function ProgIDToClassID(const ProgID: string): TGUID;
2109: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2110: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2111: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
      const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2112: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
      ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2113: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2114: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2115: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2116: Function Push( AItem : Pointer) : Pointer
2117: Function Push( AObject : TObject) : TObject
2118: Function Put1( AURL : string; const ASource : TStream) : string;
2119: Function Pythagoras( const X, Y : Extended) : Extended
2120: Function queryDLLInterface( var queryList : TStringList) : TStringList
2121: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2122: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2123: Function queryPerformanceCounter2(mse: int64): int64;
2124: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2125: //Function QueryPerformanceFrequency(mse: int64): boolean;
2126: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2127: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2128: Procedure QueryPerformanceCounter1(var aC: Int64);
2129: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2130: Function Quote( const ACommand : String) : SmallInt
2131: Function QuotedStr( S : string) : string
2132: Function RadToCycle( const Radians : Extended) : Extended
2133: Function RadToDeg( const Radians : Extended) : Extended
2134: Function RadToDeg( const Value : Extended) : Extended;
2135: Function RadToDeg1( const Value : Double) : Double;
2136: Function RadToDeg2( const Value : Single) : Single;
2137: Function RadToGrad( const Radians : Extended) : Extended
2138: Function RadToGrad( const Value : Extended) : Extended;
2139: Function RadToGrad1( const Value : Double) : Double;
2140: Function RadToGrad2( const Value : Single) : Single;
2141: Function RandG( Mean, StdDev : Extended) : Extended
2142: function Random(const ARange: Integer): Integer;
2143: function random2(a: integer): double
2144: function RandomE: Extended;
2145: function RandomF: Extended;
2146: Function RandomFrom( const AValues : array of string) : string;
2147: Function RandomRange( const AFrom, ATo : Integer) : Integer
2148: function randSeed: longint
2149: Function RawToDataColumn( ACol : Integer) : Integer
2150: Function Read : Char
2151: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2152: function Read(Buffer:String;Count:LongInt):LongInt
2153: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2154: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2155: Function ReadCardinal( const AConvert : boolean) : Cardinal
2156: Function ReadChar : Char
2157: Function ReadClient( var Buffer, Count : Integer) : Integer
2158: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2159: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2160: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2161: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:
      Bool):Int
2162: Function ReadInteger( const AConvert : boolean) : Integer
2163: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2164: Function ReadLn : string
2165: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2166: function Readln(question: string): string;
2167: Function ReadLnWait( AFailCount : Integer) : string
2168: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2169: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2170: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2171: Function ReadString( const ABytes : Integer) : string
2172: Function ReadString( const Section, Ident, Default : string) : string
2173: Function ReadString( Count : Integer) : string
2174: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2175: Function ReadTimeStampCounter : Int64
2176: Function RebootOS : Boolean
2177: Function Receive( ATimeOut : Integer) : TReplyStatus
2178: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2179: Function ReceiveLength : Integer
2180: Function ReceiveText : string
2181: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2182: Function ReceiveSerialText: string
```

```
2183: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2184: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
      AMilliSec:Word):TDateTime
2185: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2186: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2187: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2188: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2189: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2190: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2191: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2192: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2193: Function Reconcile( const Results : OleVariant) : Boolean
2194: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2195: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2196: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2197: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2198: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2199: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2200: Function RectCenter( const R : TRect) : TPoint
2201: Function RectEqual( const R1, R2 : TRect) : Boolean
2202: Function RectHeight( const R : TRect) : Integer
2203: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2204: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2205: Function RectIntersection( const R1, R2 : TRect) : TRect
2206: Function RectIntersectRect( const R1, R2 : TRect) : Boolean
2207: Function RectIsEmpty( const R : TRect) : Boolean
2208: Function RectIsNull( const R : TRect) : Boolean
2209: Function RectIsSquare( const R : TRect) : Boolean
2210: Function RectIsValid( const R : TRect) : Boolean
2211: Function RectsAreValid( R : array of TRect) : Boolean
2212: Function RectUnion( const R1, R2 : TRect) : TRect
2213: Function RectWidth( const R : TRect) : Integer
2214: Function RedComponent( const Color32 : TColor32) : Integer
2215: Function Refresh : Boolean
2216: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2217: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2218: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2219: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType
2220: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2221: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2222: Function ReleaseHandle : HBITMAP
2223: Function ReleaseHandle : HENHMETAFILE
2224: Function ReleaseHandle : HICON
2225: Function ReleasePalette : HPALETTE
2226: Function RemainderFloat( const X, Y : Float) : Float
2227: Function Remove( AClass : TClass) : Integer
2228: Function Remove( AComponent : TComponent) : Integer
2229: Function Remove( AItem : Integer) : Integer
2230: Function Remove( AItem : Pointer) : Pointer
2231: Function Remove( AItem : TObject) : TObject
2232: Function Remove( AObject : TObject) : Integer
2233: Function RemoveBackslash( const PathName : string) : string
2234: Function RemoveDF( aString : String) : String  //removes thousand separator
2235: Function RemoveDir( Dir : string) : Boolean
2236: function RemoveDir(const Dir: string): Boolean)
2237: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2238: Function RemoveFileExt( const FileName : string) : string
2239: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2240: Function RenameFile( OldName, NewName : string) : Boolean
2241: function RenameFile(const OldName: string; const NewName: string): Boolean)
2242: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2243: Function ReplaceText( const AText, AFromText, AToText : string) : string
2244: Function Replicate(c : char;I : longInt) : String;
2245: Function Request : TWebRequest
2246: Function ResemblesText( const AText, AOther : string) : Boolean
2247: Function Reset : Boolean
2248: function Reset2(mypath: string):string;
2249: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2250: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2251: Function Response : TWebResponse
2252: Function ResumeSupported : Boolean
2253: Function RETHINKHOTKEYS : BOOLEAN
2254: Function RETHINKLINES : BOOLEAN
2255: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2256: Function RetrieveCurrentDir : string
2257: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2258: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2259: Function RetrieveMailBoxSize : integer
2260: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2261: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2262: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2263: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2264: Function ReverseBits( Value : Byte) : Byte;
2265: Function ReverseBits1( Value : Shortint) : Shortint;
2266: Function ReverseBits2( Value : Smallint) : Smallint;
2267: Function ReverseBits3( Value : Word) : Word;
2268: Function ReverseBits4( Value : Cardinal) : Cardinal;
2269: Function ReverseBits4( Value : Integer) : Integer;
2270: Function ReverseBits5( Value : Int64) : Int64;
```

```
2271: Function ReverseBytes( Value : Word) : Word;
2272: Function ReverseBytes1( Value : Smallint) : Smallint;
2273: Function ReverseBytes2( Value : Integer) : Integer;
2274: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2275: Function ReverseBytes4( Value : Int64) : Int64;
2276: Function ReverseString( const AText : string) : string
2277: Function ReverseDNSLookup(const IPAddrs:String;DNSServer:String;Timeout,Retries:Int;var
      HostName:String):Bool;
2278: Function Revert : HResult
2279: Function RGB(R,G,B: Byte): TColor;
2280: Function RGB2BGR( const Color : TColor) : TColor
2281: Function RGB2TColor( R, G, B : Byte) : TColor
2282: Function RGBToWebColorName( RGB : Integer) : string
2283: Function RGBToWebColorStr( RGB : Integer) : string
2284: Function RgbToHtml( Value : TColor) : string
2285: Function HtmlToRgb(const Value: string): TColor;
2286: Function RightStr( const AStr : String; Len : Integer) : String
2287: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2288: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2289: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2290: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
2291: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2292: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2293: Function Round(e : Extended) : Longint;
2294: Function Round64(e: extended): Int64;
2295: Function RoundAt( const Value : string; Position : SmallInt) : string
2296: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2297: Function RoundTo(const AValue: Extended; const ADigit: TRoundToEXRangeExtended): Extended;');
2298: Function SimpleRoundTo(const AValue: Extended; const ADigit: TRoundToRange): Extended;');
2299: Function RoundFrequency( const Frequency : Integer) : Integer
2300: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2301: Function RoundPoint( const X, Y : Double) : TPoint
2302: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2303: Function RowCount : Integer
2304: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2305: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2306: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2307: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2308: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2309: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2310: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2311: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2312: Function S_AddBackSlash( const ADirName : string) : string
2313: Function S_AllTrim( const cStr : string) : string
2314: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2315: Function S_Cut( const cStr : string; const iLen : integer) : string
2316: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2317: Function S_DirExists( const ADir : string) : Boolean
2318: Function S_Empty( const cStr : string) : boolean
2319: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2320: Function S_LargeFontsActive : Boolean
2321: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2322: Function S_LTrim( const cStr : string) : string
2323: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2324: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2325: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2326: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2327: Function S_RTrim( const cStr : string) : string
2328: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2329: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2330: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2331: Function S_Space( const iLen : integer) : String
2332: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2333: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2334: Function S_StrCRC32( const Text : string) : LongWORD
2335: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2336: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2337: Function S_StringtoUTF_8( const AString : string) : string
2338: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2339: function S_StrToReal(const cStr: string; var R: Double): Boolean
2340: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2341: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2342: Function S_UTF_8ToString( const AString : string) : string
2343: Function S_WBox( const AText : string) : integer
2344: Function SameDate( const A, B : TDateTime) : Boolean
2345: function SameDate(const A, B: TDateTime): Boolean;
2346: Function SameDateTime( const A, B : TDateTime) : Boolean
2347: function SameDateTime(const A, B: TDateTime): Boolean;
2348: Function SameFileName( S1, S2 : string) : Boolean
2349: Function SameText( S1, S2 : string) : Boolean
2350: function SameText(const S1: string; const S2: string): Boolean)
2351: Function SameTime( const A, B : TDateTime) : Boolean
2352: function SameTime(const A, B: TDateTime): Boolean;
2353: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2354: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2355: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2356: Function SampleVariance( const X : TDynFloatArray) : Float
2357: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2358: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
```

```
2359: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2360: Function SaveToFile( const AFileName : TFileName) : Boolean
2361: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2362: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2363: Function ScanF(const aformat: String; const args: array of const): string;
2364: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2365: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
      TStringSearchOptions):PChar
2366: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2367: function SearchRecattr: integer;
2368: function SearchRecExcludeAttr: integer;
2369: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2370: function SearchRecname: string;
2371: function SearchRecsize: integer;
2372: function SearchRecTime: integer;
2373: Function Sec( const X : Extended) : Extended
2374: Function Secant( const X : Extended) : Extended
2375: Function SecH( const X : Extended) : Extended
2376: Function SecondOf( const AValue : TDateTime) : Word
2377: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2378: Function SecondOfTheHour( const AValue : TDateTime) : Word
2379: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2380: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2381: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2382: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2383: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2384: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2385: Function SectionExists( const Section : string) : Boolean
2386: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2387: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2388: function Seek(Offset:LongInt;Origin:Word):LongInt
2389: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2390: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2391: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2392: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2393: Function SendBuf( var Buf, Count : Integer) : Integer
2394: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2395: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2396: Function SendKey( AppName : string; Key : Char) : Boolean
2397: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2398: Function SendStream( AStream : TStream) : Boolean
2399: Function SendStreamThenDrop( AStream : TStream) : Boolean
2400: Function SendText( const S : string) : Integer
2401: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2402: Function SendSerialText(Data: String): cardinal
2403: Function Sent : Boolean
2404: Function ServicesFilePath: string
2405: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2406: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2407: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2408: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2409: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2410: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2411: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2412: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2413: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2414: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2415: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2416: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2417: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2418: Function SetCurrentDir( Dir : string) : Boolean
2419: function SetCurrentDir(const Dir: string): Boolean)
2420: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2421: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2422: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2423: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2424: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2425: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2426: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2427: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2428: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2429: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2430: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2431: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2432: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2433: Function SetLocalTime( Value : TDateTime) : boolean
2434: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2435: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2436: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2437: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2438: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2439: Function SetSize( libNewSize : Longint) : HResult
2440: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2441: Function Sgn( const X : Extended) : Integer
2442: function SHA1(const fileName: string): string;
2443: function SHA256(astr: string; amode: char): string)
2444: function SHA512(astr: string; amode: char): string)
```

```
2445: Function ShareMemoryManager : Boolean
2446: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2447: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2448: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2449: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2450: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2451: function ShortDateFormat: string;
2452: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2453: function ShortTimeFormat: string;
2454: function SHOWMODAL:INTEGER
2455: Function ShowModalControl(aControl : TControl; BS : TFormBorderStyle; BI : TBorderIcons; WS :
      TWindowState; aColor : TColor; BW : Integer; Title : String; BeforeShowModal : TNotifyEvent) :
      TModalResult');
2456: Function
      ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2457: function ShowWindow(C1: HWND; C2: integer): boolean;
2458: procedure ShowMemory     //in Dialog
2459: function ShowMemory2: string;
2460: Function ShutDownOS : Boolean
2461: Function Signe( const X, Y : Extended) : Extended
2462: Function Sign( const X : Extended) : Integer
2463: Function Sin(e : Extended) : Extended;
2464: Function sinc( const x : Double) : Double
2465: Function SinJ( X : Float) : Float
2466: Function Size( const AFileName : String) : Integer
2467: function SizeOf: Longint;
2468: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2469: function SlashSep(const Path, S: String): String
2470: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2471: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2472: Function SmallPoint(X, Y: Integer): TSmallPoint)
2473: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2474: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2475: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2476: Function SoundexProc( const AText, AOther : string) : Boolean
2477: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2478: Function SoundexWord( const AText : string) : Word
2479: Function SourcePos : Longint
2480: function SourcePos:LongInt
2481: Function Split0( Str : string; const substr : string) : TStringList
2482: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2483: Function SQLRequiresParams( const SQL : WideString) : Boolean
2484: Function Sqr(e : Extended) : Extended;
2485: Function Sqrt(e : Extended) : Extended;
2486: Function StartIP : String
2487: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2488: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2489: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2490: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2491: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2492: Function StartOfAYear( const AYear : Word) : TDateTime
2493: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2494: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2495: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2496: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2497: Function StartsStr( const ASubText, AText : string) : Boolean
2498: Function StartsText( const ASubText, AText : string) : Boolean
2499: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2500: Function StartsWith( const str : string; const sub : string) : Boolean
2501: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2502: Function StatusString( StatusCode : Integer) : string
2503: Function StdDev( const Data : array of Double) : Extended
2504: Function Stop : Float
2505: Function StopCount( var Counter : TJclCounter) : Float
2506: Function StoreColumns : Boolean
2507: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2508: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2509: Function StrAlloc( Size : Cardinal) : PChar
2510: function StrAlloc(Size: Cardinal): PChar)
2511: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2512: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2513: Function StrBufSize( Str : PChar) : Cardinal
2514: function StrBufSize(const Str: PChar): Cardinal)
2515: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2516: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2517: Function StrCat( Dest : PChar; Source : PChar) : PChar
2518: function StrCat(Dest: PChar; const Source: PChar): PChar)
2519: Function StrCharLength( Str : PChar) : Integer
2520: Function StrComp( Str1, Str2 : PChar) : Integer
2521: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2522: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2523: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2524: Function Stream_to_AnsiString( Source : TStream) : ansistring
2525: Function Stream_to_Base64( Source : TStream) : ansistring
2526: Function Stream_to_decimalbytes( Source : TStream) : string
2527: Function Stream2WideString( oStream : TStream) : WideString
2528: Function StreamtoAnsiString( Source : TStream) : ansistring
2529: Function StreamToByte( Source : TStream) : string
```

```
2530: Function StreamToDecimalbytes( Source : TStream) : string
2531: Function StreamtoOrd( Source : TStream) : string
2532: Function StreamToString( Source : TStream) : string
2533: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2534: Function StrEmpty( const sString : string) : boolean
2535: Function StrEnd( Str : PChar) : PChar
2536: function StrEnd(const Str: PChar): PChar)
2537: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2538: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2539: Function StrGet(var S : String; I : Integer) : Char;
2540: Function StrGet2(S : String; I : Integer) : Char;
2541: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2542: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2543: Function StrHtmlDecode( const AStr : String) : String
2544: Function StrHtmlEncode( const AStr : String) : String
2545: Function StrToBytes(const Value: String): TBytes;
2546: Function StrIComp( Str1, Str2 : PChar) : Integer
2547: Function StringOfChar(c : char;I : longInt) : String;
2548: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2549: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2550: Function StringRefCount(const s: String): integer;
2551: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2552: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2553: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2554: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2555: Function StringToBoolean( const Ps : string) : Boolean
2556: function StringToColor(const S: string): TColor)
2557: Function StringToCursor(const S: string): TCursor;
2558: function StringToGUID(const S: string): TGUID)
2559: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2560: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2561: Function StringWidth( S : string) : Integer
2562: Function StrInternetToDateTime( Value : string) : TDateTime
2563: Function StrIsDateTime( const Ps : string) : Boolean
2564: Function StrIsFloatMoney( const Ps : string) : Boolean
2565: Function StrIsInteger( const S : string) : Boolean
2566: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2567: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2568: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2569: Function StrLen( Str : PChar) : Cardinal
2570: function StrLen(const Str: PChar): Cardinal)
2571: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2572: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2573: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2574: Function StrLower( Str : PChar) : PChar
2575: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2576: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2577: Function StrNew( Str : PChar) : PChar
2578: function StrNew(const Str: PChar): PChar)
2579: Function StrNextChar( Str : PChar) : PChar
2580: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2581: Function StrParse( var sString : string; const sDelimiters : string) : string;
2582: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2583: Function StrPas( Str : PChar) : string
2584: function StrPas(const Str: PChar): string)
2585: Function StrPCopy( Dest : PChar; Source : string) : PChar
2586: function StrPCopy(Dest: PChar; const Source: string): PChar)
2587: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2588: Function StrPos( Str1, Str2 : PChar) : PChar
2589: Function StrScan(const Str: PChar; Chr: Char): PChar)
2590: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2591: Function StrToBcd( const AValue : string) : TBcd
2592: Function StrToBool( S : string) : Boolean
2593: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2594: Function StrToCard( const AStr : String) : Cardinal
2595: Function StrToConv( AText : string; out AType : TConvType) : Double
2596: Function StrToCurr( S : string) : Currency;
2597: function StrToCurr(const S: string): Currency)
2598: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2599: Function StrToDate( S : string) : TDateTime;
2600: function StrToDate(const s: string): TDateTime;
2601: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2602: Function StrToDateTime( S : string) : TDateTime;
2603: function StrToDateTime(const S: string): TDateTime)
2604: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2605: Function StrToDay( const ADay : string) : Byte
2606: Function StrToFloat( S : string) : Extended;
2607: function StrToFloat(s: String): Extended;
2608: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2609: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2610: Function StrToFloat( S : string) : Extended;
2611: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2612: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2613: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2614: Function StrToCurr( S : string) : Currency;
2615: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2616: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2617: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2618: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
```

```
2619: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2620: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2621: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2622: Function StrToDateTime( S : string) : TDateTime;
2623: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2624: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2625: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2626: Function StrToInt( S : string) : Integer
2627: function StrToInt(s: String): Longint;
2628: Function StrToInt64( S : string) : Int64
2629: function StrToInt64(s: String): int64;
2630: Function StrToInt64Def( S : string; Default : Int64) : Int64
2631: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2632: Function StrToIntDef( S : string; Default : Integer) : Integer
2633: function StrToIntDef(const S: string; Default: Integer): Integer)
2634: Function StrToIntDef(s: String; def: Longint): Longint;
2635: Function StrToMonth( const AMonth : string) : Byte
2636: Function StrToTime( S : string) : TDateTime;
2637: function StrToTime(const S: string): TDateTime)
2638: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2639: Function StrToWord( const Value : String) : Word
2640: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2641: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2642: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2643: Function StrUpper( Str : PChar) : PChar
2644: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2645: Function Sum( const Data : array of Double) : Extended
2646: Function SumFloatArray( const B : TDynFloatArray) : Float
2647: Function SumInt( const Data : array of Integer) : Integer
2648: Function SumOfSquares( const Data : array of Double) : Extended
2649: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2650: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2651: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2652: Function Supports( CursorOptions : TCursorOptions) : Boolean
2653: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2654: Function SwapWord(w : word): word)
2655: Function SwapInt(i : integer): integer)
2656: Function SwapLong(L : longint): longint)
2657: Function Swap(i : integer): integer)
2658: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2659: Function SyncTime : Boolean
2660: Function SysErrorMessage( ErrorCode : Integer) : string
2661: function SysErrorMessage(ErrorCode: Integer): string)
2662: Function SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2663: function SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2664: Function SysStringLen(const S: WideString): Integer; stdcall;
2665: Function TabRect( Index : Integer) : TRect
2666: Function Tan( const X : Extended) : Extended
2667: Function TaskMessageDlg(const Title,
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2668: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2669: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2670: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2671: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2672: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2673: Function TenToY( const Y : Float) : Float
2674: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2675: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2676: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2677: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2678: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2679: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2680: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2681: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2682: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2683: Function TestBits( const Value, Mask : Byte) : Boolean;
2684: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2685: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2686: Function TestBits3( const Value, Mask : Word) : Boolean;
2687: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2688: Function TestBits5( const Value, Mask : Integer) : Boolean;
2689: Function TestBits6( const Value, Mask : Int64) : Boolean;
2690: Function TestFDIVInstruction : Boolean
2691: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2692: Function TextExtent( const Text : string) : TSize
2693: function TextHeight(Text: string): Integer;
2694: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2695: Function TextStartsWith( const S, SubS : string) : Boolean
2696: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2697: Function ConvInteger(i : integer):string;
2698: Function IntegerToText(i : integer):string;
2699: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2700: function TextWidth(Text: string): Integer;
2701: Function ThreadCount : integer
```

```
2702: function ThousandSeparator: char;
2703: Function Ticks : Cardinal
2704: Function Time : TDateTime
2705: function Time: TDateTime;
2706: function TimeGetTime: int64;
2707: Function TimeOf( const AValue : TDateTime) : TDateTime
2708: function TimeSeparator: char;
2709: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2710: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2711: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2712: Function TimeToStr( DateTime : TDateTime) : string;
2713: function TimeToStr(const DateTime: TDateTime): string;
2714: Function TimeZoneBias : TDateTime
2715: Function ToCommon( const AValue : Double) : Double
2716: function ToCommon(const AValue: Double): Double;
2717: Function Today : TDateTime
2718: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2719: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2720: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2721: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2722: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2723: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2724: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2725: function TokenComponentIdent:String
2726: Function TokenFloat : Extended
2727: function TokenFloat:Extended
2728: Function TokenInt : Longint
2729: function TokenInt:LongInt
2730: Function TokenString : string
2731: function TokenString:String
2732: Function TokenSymbolIs( const S : string) : Boolean
2733: function TokenSymbolIs(S:String):Boolean
2734: Function Tomorrow : TDateTime
2735: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2736: Function ToString : string
2737: Function TotalVariance( const Data : array of Double) : Extended
2738: Function Trace2( AURL : string) : string;
2739: Function TrackMenu( Button : TToolButton) : Boolean
2740: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2741: Function TranslateURI( const URI : string) : string
2742: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2743: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
     SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2744: Function Trim( S : string) : string;
2745: Function Trim( S : WideString) : WideString;
2746: Function Trim(s : AnyString) : AnyString;
2747: Function TrimAllOf( ATrim, AText : String) : String
2748: Function TrimLeft( S : string) : string;
2749: Function TrimLeft( S : WideString) : WideString;
2750: function TrimLeft(const S: string): string)
2751: Function TrimRight( S : string) : string;
2752: Function TrimRight( S : WideString) : WideString;
2753: function TrimRight(const S: string): string)
2754: function TrueBoolStrs: array of string
2755: Function Trunc(e : Extended) : Longint;
2756: Function Trunc64(e: extended): Int64;
2757: Function TruncPower( const Base, Exponent : Float) : Float
2758: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2759: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2760: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2761: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2762: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2763: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
     AValue:TDateTime):Boolean
2764: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2765: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
     AVal:TDateTime):Bool
2766: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2767: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2768: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2769: Function TryLock : Boolean
2770: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2771: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
     AMilliSecond : Word; out AResult : TDateTime) : Boolean
2772: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2773: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2774: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2775: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2776: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2777: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2778: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2779: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2780: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2781: Function TwoToY( const Y : Float) : Float
2782: Function UCS4StringToWideString( const S : UCS4String) : WideString
2783: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2784: function Unassigned: Variant;
2785: Function UndoLastChange( FollowChange : Boolean) : Boolean
2786: function UniCodeToStr(Value: string): string;
```

```
2787: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2788: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2789: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2790: Function UnixPathToDosPath( const Path : string) : string
2791: Function UnixToDateTime( const AValue : Int64) : TDateTime
2792: function UnixToDateTime(U: Int64): TDateTime;
2793: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2794: Function UnlockResource( ResData : HGLOBAL) : LongBool
2795: Function UnlockVolume( var Handle : THandle) : Boolean
2796: Function UnMaskString( Mask, Value : String) : String
2797: function UpCase(ch : Char ) : Char;
2798: Function UpCaseFirst( const AStr : string) : string
2799: Function UpCaseFirstWord( const AStr : string) : string
2800: Function UpdateAction( Action : TBasicAction) : Boolean
2801: Function UpdateKind : TUpdateKind
2802: Function UPDATESTATUS : TUPDATESTATUS
2803: Function UpperCase( S : string) : string
2804: Function Uppercase(s : AnyString) : AnyString;
2805: Function URLDecode( ASrc : string) : string
2806: Function URLEncode( const ASrc : string) : string
2807: Function UseRightToLeftAlignment : Boolean
2808: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2809: Function UseRightToLeftReading : Boolean
2810: Function UTF8CharLength( Lead : Char) : Integer
2811: Function UTF8CharSize( Lead : Char) : Integer
2812: Function UTF8Decode( const S : UTF8String) : WideString
2813: Function UTF8Encode( const WS : WideString) : UTF8String
2814: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2815: Function Utf8ToAnsi( const S : UTF8String) : string
2816: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2817: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2818: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2819: Function ValidParentForm(control: TControl): TForm
2820: Function Value : Variant
2821: Function ValueExists( const Section, Ident : string) : Boolean
2822: Function ValueOf( const Key : string) : Integer
2823: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2824: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2825: Function VarArrayFromStrings( Strings : TStrings) : Variant
2826: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2827: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2828: Function VarFMTBcd : TVarType
2829: Function VarFMTBcdCreate1 : Variant;
2830: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2831: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2832: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2833: Function Variance( const Data : array of Double) : Extended
2834: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2835: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2836: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2837: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2838: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2839: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2840: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2841: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2842: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2843: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2844: Function VariantNeg( const V1 : Variant) : Variant
2845: Function VariantNot( const V1 : Variant) : Variant
2846: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2847: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2848: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2849: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2850: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2851: function VarIsEmpty(const V: Variant): Boolean;
2852: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2853: function VarIsNull(const V: Variant): Boolean;
2854: Function VarToBcd( const AValue : Variant) : TBcd
2855: function VarType(const V: Variant): TVarType;
2856: Function VarType( const V : Variant) : TVarType
2857: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2858: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2859: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2860: Function VarIsByRef( const V : Variant) : Boolean
2861: Function VarIsEmpty( const V : Variant) : Boolean
2862: Procedure VarCheckEmpty( const V : Variant)
2863: Function VarIsNull( const V : Variant) : Boolean
2864: Function VarIsClear( const V : Variant) : Boolean
2865: Function VarIsCustom( const V : Variant) : Boolean
2866: Function VarIsOrdinal( const V : Variant) : Boolean
2867: Function VarIsFloat( const V : Variant) : Boolean
2868: Function VarIsNumeric( const V : Variant) : Boolean
2869: Function VarIsStr( const V : Variant) : Boolean
2870: Function VarToStr( const V : Variant) : string
2871: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2872: Function VarToWideStr( const V : Variant) : WideString
2873: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2874: Function VarToDateTime( const V : Variant) : TDateTime
2875: Function VarFromDateTime( const DateTime : TDateTime) : Variant
```

```
2876: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2877: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2878: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2879: Function VarSameValue( const A, B : Variant) : Boolean
2880: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2881: Function VarIsEmptyParam( const V : Variant) : Boolean
2882: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2883: Function VarIsError1( const V : Variant) : Boolean;
2884: Function VarAsError( AResult : HRESULT) : Variant
2885: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2886: Function VarIsArray( const A : Variant) : Boolean;
2887: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2888: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2889: Function VarArrayOf( const Values : array of Variant) : Variant
2890: Function VarArrayRef( const A : Variant) : Variant
2891: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2892: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2893: Function VarArrayDimCount( const A : Variant) : Integer
2894: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2895: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2896: Function VarArrayLock( const A : Variant) : ___Pointer
2897: Procedure VarArrayUnlock( const A : Variant)
2898: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2899: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2900: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)
2901: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)
2902: Function Unassigned : Variant
2903: Function Null : Variant
2904: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2905: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2906: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2907: function VectorDot(const V1,V2: TFloatPoint): Double;
2908: Function VectorLengthSqr( const V : TFloatPoint) : Double
2909: function VectorLengthSqr(const V: TFloatPoint): Double;
2910: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
2911: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2912: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2913: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2914: Function Verify( AUserName : String) : String
2915: Function Versine( X : Float) : Float
2916: function VersionCheck: boolean;
2917: function VersionCheckAct: string;
2918: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2919: Function VersionLanguageName( const LangId : Word) : string
2920: Function VersionResourceAvailable( const FileName : string) : Boolean
2921: Function Visible : Boolean
2922: function VolumeID(DriveChar: Char): string
2923: Function WaitFor( const AString : string) : string
2924: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
2925: Function WaitFor1 : TWaitResult;
2926: Function WaitForData( Timeout : Longint) : Boolean
2927: Function WebColorNameToColor( WebColorName : string) : TColor
2928: Function WebColorStrToColor( WebColor : string) : TColor
2929: Function WebColorToRGB( WebColor : Integer) : Integer
2930: Function wGet(aURL, afile: string): boolean;'
2931: Function wGet2(aURL, afile: string): boolean;'  //without file open
2932: Function WebGet(aURL, afile: string): boolean;'
2933: Function WebExists: boolean;   //alias to isinternet
2934: Function WeekOf( const AValue : TDateTime) : Word
2935: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2936: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
2937: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2938: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2939: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
2940: Function WeeksInAYear( const AYear : Word) : Word
2941: Function WeeksInYear( const AValue : TDateTime) : Word
2942: Function WeekSpan( const ANow, AThen : TDateTime) : Double
2943: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString
2944: Function WideCat( const x, y : WideString) : WideString
2945: Function WideCompareStr( S1, S2 : WideString) : Integer
2946: function WideCompareStr(const S1: WideString; const S2: WideString): Integer)
2947: Function WideCompareText( S1, S2 : WideString) : Integer
2948: function WideCompareText(const S1: WideString; const S2: WideString): Integer)
2949: Function WideCopy( const src : WideString; index, count : Integer) : WideString
2950: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
2951: Function WideEqual( const x, y : WideString) : Boolean
2952: function WideFormat(const Format: WideString; const Args: array of const): WideString)
2953: Function WideGreater( const x, y : WideString) : Boolean
2954: Function WideLength( const src : WideString) : Integer
2955: Function WideLess( const x, y : WideString) : Boolean
2956: Function WideLowerCase( S : WideString) : WideString
2957: function WideLowerCase(const S: WideString): WideString)
2958: Function WidePos( const src, sub : WideString) : Integer
2959: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
2960: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
2961: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
2962: Function WideSameStr( S1, S2 : WideString) : Boolean
2963: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
2964: Function WideSameText( S1, S2 : WideString) : Boolean
```

```
2965: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
2966: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
2967: Function WideStringToUCS4String( const S : WideString) : UCS4String
2968: Function WideUpperCase( S : WideString) : WideString
2969: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
2970: function Win32Check(RetVal: boolean): boolean)
2971: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
2972: Function Win32RestoreFile( const FileName : string) : Boolean
2973: Function Win32Type : TIdWin32Type
2974: Function WinColor( const Color32 : TColor32) : TColor
2975: function winexec(FileName: pchar; showCmd: integer): integer;
2976: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
2977: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
2978: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
2979: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
2980: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
2981: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
2982: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
2983: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
2984: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
2985: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
2986: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
2987: Function WordToStr( const Value : Word) : String
2988: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
2989: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
2990: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
2991: Function WorkArea : Integer
2992: Function WrapText( Line : string; MaxCol : Integer) : string;
2993: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
2994: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
2995: function Write(Buffer:String;Count:LongInt):LongInt
2996: Function WriteClient( var Buffer, Count : Integer) : Integer
2997: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
2998: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
2999: Function WriteString( const AString : string) : Boolean
3000: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
3001: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
3002: Function wsprintf( Output : PChar; Format : PChar) : Integer
3003: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
3004: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
3005: Function XorDecode( const Key, Source : string) : string
3006: Function XorEncode( const Key, Source : string) : string
3007: Function XorString( const Key, Src : ShortString) : ShortString
3008: Function Yield : Bool
3009: Function YearOf( const AValue : TDateTime) : Word
3010: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3011: Function YearSpan( const ANow, AThen : TDateTime) : Double
3012: Function Yesterday : TDateTime
3013: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3014: Function( const Name : string; Proc : TUserFunction)
3015: Function using Special_Scholz from 3.8.5.0
3016: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3017: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3018: Function FloatToTime2Dec(value:Extended):Extended;
3019: Function MinToStd(value:Extended):Extended;
3020: Function MinToStdAsString(value:Extended):String;
3021: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3022: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3023: Function Round2Dec (zahl:Extended):Extended;
3024: Function GetAngle(x,y:Extended):Double;
3025: Function AddAngle(a1,a2:Double):Double;
3026:
3027: ************************************************************************
3028: unit uPSI_StText;
3029: ************************************************************************
3030: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3031: Function TextFileSize( var F : TextFile) : LongInt
3032: Function TextPos( var F : TextFile) : LongInt
3033: Function TextFlush( var F : TextFile) : Boolean
3034:
3035: ************************************************************************
3036: from  JvVCLUtils;
3037: ************************************************************************
3038: { Windows resources (bitmaps and icons) VCL-oriented routines }
3039: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3040: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
     DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3041: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
     Bitmap: TBitmap; TransparentColor: TColor);
3042: function MakeBitmap(ResID: PChar): TBitmap;
3043: function MakeBitmapID(ResID: Word): TBitmap;
3044: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3045: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3046: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3047: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3048:   HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3049: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3050: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3051: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
```

```
3052: {$IFDEF WIN32}
3053: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3054:   X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3055: {$ENDIF}
3056: function MakeIcon(ResID: PChar): TIcon;
3057: function MakeIconID(ResID: Word): TIcon;
3058: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3059: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3060: {$IFDEF WIN32}
3061: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3062: {$ENDIF}
3063: { Service routines }
3064: procedure NotImplemented;
3065: procedure ResourceNotFound(ResID: PChar);
3066: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3067: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3068: function PaletteColor(Color: TColor): Longint;
3069: function WidthOf(R: TRect): Integer;
3070: function HeightOf(R: TRect): Integer;
3071: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3072: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3073: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3074: procedure Delay(MSecs: Longint);
3075: procedure CenterControl(Control: TControl);
3076: Function PaletteEntries( Palette : HPALETTE) : Integer
3077: Function WindowClassName( Wnd : HWND) : string
3078: Function ScreenWorkArea : TRect
3079: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3080: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3081: Procedure ActivateWindow( Wnd : HWND)
3082: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3083: Procedure CenterWindow( Wnd : HWND)
3084: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3085: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3086: Function DialogsToPixelsX( Dlgs : Word) : Word
3087: Function DialogsToPixelsY( Dlgs : Word) : Word
3088: Function PixelsToDialogsX( Pixs : Word) : Word
3089: Function PixelsToDialogsY( Pixs : Word) : Word
3090: {$IFDEF WIN32}
3091: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3092: function MakeVariant(const Values: array of Variant): Variant;
3093: {$ENDIF}
3094: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3095: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3096: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3097: {$IFDEF CBUILDER}
3098: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3099: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3100: {$ELSE}
3101: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3102: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3103: {$ENDIF CBUILDER}
3104: function IsForegroundTask: Boolean;
3105: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3106: function GetAveCharSize(Canvas: TCanvas): TPoint;
3107: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3108: procedure FreeUnusedOle;
3109: procedure Beep;
3110: function GetWindowsVersionJ: string;
3111: function LoadDLL(const LibName: string): THandle;
3112: function RegisterServer(const ModuleName: string): Boolean;
3113: {$IFNDEF WIN32}
3114: function IsLibrary: Boolean;
3115: {$ENDIF}
3116: { Gradient filling routine }
3117: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3118: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
     TFillDirection; Colors: Byte);
3119: { String routines }
3120: function GetEnvVar(const VarName: string): string;
3121: function AnsiUpperFirstChar(const S: string): string;
3122: function StringToPChar(var S: string): PChar;
3123: function StrPAlloc(const S: string): PChar;
3124: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3125: function DropT(const S: string): string;
3126: { Memory routines }
3127: function AllocMemo(Size: Longint): Pointer;
3128: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3129: procedure FreeMemo(var fpBlock: Pointer);
3130: function GetMemoSize(fpBlock: Pointer): Longint;
3131: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3132: {$IFNDEF COMPILER5_UP}
3133: procedure FreeAndNil(var Obj);
3134: {$ENDIF}
3135: // from PNGLoader
3136: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3137: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3138: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3139: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
```

```
3140: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
      TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
3141:   Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3142:   Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3143:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3144:  //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3145:  //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3146:  //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3147:  Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3148:  Procedure SetImeName( Name : TImeName)
3149:  Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3150:  Function Imm32GetContext( hWnd : HWND) : HIMC
3151:  Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
3152:  Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
3153:  Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3154:  Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3155: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3156:  //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
3157:  Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3158:  Function Imm32IsIME( hKl : longword) : Boolean
3159:  Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3160:  Procedure DragDone( Drop : Boolean)
3161:
3162:
3163: //****************************************added  from jvjvclutils
3164: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3165: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3166: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3167: function IsPositiveResult(Value: TModalResult): Boolean;
3168: function IsNegativeResult(Value: TModalResult): Boolean;
3169: function IsAbortResult(const Value: TModalResult): Boolean;
3170: function StripAllFromResult(const Value: TModalResult): TModalResult;
3171: // returns either BrightColor or DarkColor depending on the luminance of AColor
3172: // This function gives the same result (AFAIK) as the function used in Windows to
3173: // calculate the desktop icon text color based on the desktop background color
3174: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3175: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3176:
3177: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3178:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3179:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3180:   var LinkName: string; Scale: Integer = 100); overload;
3181: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3182:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3183:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3184:   var LinkName: string; Scale: Integer = 100); overload;
3185: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3186:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3187: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3188:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3189:   Scale: Integer = 100): string;
3190: function HTMLPlainText(const Text: string): string;
3191: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3192:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3193: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3194:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3195: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3196: function HTMLPrepareText(const Text: string): string;
3197:
3198: ************************************** uPSI_JvAppUtils;
3199: Function GetDefaultSection( Component : TComponent) : string
3200: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3201: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3202: Function GetDefaultIniName : string
3203:  //'OnGetDefaultIniName','TOnGetDefaultIniName);
3204: Function GetDefaultIniRegKey : string
3205: Function FindForm( FormClass : TFormClass) : TForm
3206: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3207: Function ShowDialog( FormClass : TFormClass) : Boolean
3208:  //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3209: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3210: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3211: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3212: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3213: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3214: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3215: Function StrToIniStr( const Str : string) : string
3216: Function IniStrToStr( const Str : string) : string
3217: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string) : string
3218: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string)
3219: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint) : Longint
3220: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint)
3221: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean) : Boolean
3222: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean)
3223: Procedure IniReadSections( IniFile : TObject; Strings : TStrings)
3224: Procedure IniEraseSection( IniFile : TObject; const Section : string)
3225: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string)
3226: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3227: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
```

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

```
3315:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3316:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3317:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3318:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3319:  Function HoursToMSecs( Hours : Integer) : Integer
3320:  Function MinutesToMSecs( Minutes : Integer) : Integer
3321:  Function SecondsToMSecs( Seconds : Integer) : Integer
3322:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3323:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3324:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3325:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3326:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3327:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3328:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3329:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3330:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3331:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3332:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3333:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3334:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3335:  Function DosDateTimeToStr( DateTime : Integer) : string
3336:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3337:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3338:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3339:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3340:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3341:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3342:  Function FileTimeToStr( const FileTime : TFileTime) : string
3343:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3344:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3345:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3346:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3347:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3348:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3349:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3350:   TJclUnixTime32', 'Longword
3351:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3352:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3353:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3354:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3355:  Function JNullStamp : TTimeStamp
3356:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3357:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3358:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3359:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3360:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3361:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3362:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3363:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3364:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3365:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3366:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3367:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3368:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3369:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3370:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3371:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3372:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3373:   FindClass('TOBJECT'),'EJclDateTimeError
3374: end;
3375:
3376: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3377: begin
3378:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3379:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3380:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3381:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3382:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3383:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3384:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3385:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3386:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3387:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3388:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3389:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3390:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3391:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3392:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3393:  Function AbortShutDown : Boolean;
3394:  Function AbortShutDown1( const MachineName : string) : Boolean;
3395:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3396:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3397:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3398:   FindClass('TOBJECT'),'EJclCreateProcessError
3399:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3400:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
      Environment:PChar);
3401:   // with Add(EJclCreateProcessError) do
3402:  end;
```

```
3403:
3404:
3405: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3406: begin
3407:   //'AnsiSigns','Set').SetSet(['-', '+']);
3408:  'C1_UPPER','LongWord( $0001);
3409:  'C1_LOWER','LongWord( $0002);
3410:  'C1_DIGIT','LongWord').SetUInt( $0004);
3411:  'C1_SPACE','LongWord').SetUInt( $0008);
3412:  'C1_PUNCT','LongWord').SetUInt( $0010);
3413:  'C1_CNTRL','LongWord').SetUInt( $0020);
3414:  'C1_BLANK','LongWord').SetUInt( $0040);
3415:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3416:  'C1_ALPHA','LongWord').SetUInt( $0100);
3417:   AnsiChar', 'Char
3418:  Function StrIsAlpha( const S : AnsiString) : Boolean
3419:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3420:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3421:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3422:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3423:  Function StrIsDigit( const S : AnsiString) : Boolean
3424:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3425:  Function StrSame( const S1, S2 : AnsiString) : Boolean
3426:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3427:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3428:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3429:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3430:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3431:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3432:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3433:  Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
3434:  Function StrEscapedToString( const S : AnsiString) : AnsiString
3435:  Function JStrLower( const S : AnsiString) : AnsiString
3436:  Procedure StrLowerInPlace( var S : AnsiString)
3437:  ///Procedure StrLowerBuff( S : PAnsiChar)
3438:  Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3439:  Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3440:  Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3441:  Function StrProper( const S : AnsiString) : AnsiString
3442:  //Procedure StrProperBuff( S : PAnsiChar)
3443:  Function StrQuote( const S : AnsiString; C : AnsiChar) : AnsiString
3444:  Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3445:  Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3446:  Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3447:  Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar) : AnsiString
3448:  Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3449:  Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3450:  Function StrRepeat( const S : AnsiString; Count : Integer) : AnsiString
3451:  Function StrRepeatLength( const S : AnsiString; const L : Integer) : AnsiString
3452:  Function StrReverse( const S : AnsiString) : AnsiString
3453:  Procedure StrReverseInPlace( var S : AnsiString)
3454:  Function StrSingleQuote( const S : AnsiString) : AnsiString
3455:  Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet) : AnsiString
3456:  Function StrStringToEscaped( const S : AnsiString) : AnsiString
3457:  Function StrStripNonNumberChars( const S : AnsiString) : AnsiString
3458:  Function StrToHex( const Source : AnsiString) : AnsiString
3459:  Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar) : AnsiString
3460:  Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3461:  Function StrTrimCharRight( const S : AnsiString; C : AnsiChar) : AnsiString
3462:  Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3463:  Function StrTrimQuotes( const S : AnsiString) : AnsiString
3464:  Function JStrUpper( const S : AnsiString) : AnsiString
3465:  Procedure StrUpperInPlace( var S : AnsiString)
3466:  //Procedure StrUpperBuff( S : PAnsiChar)
3467:  Function StrOemToAnsi( const S : AnsiString) : AnsiString
3468:  Function StrAnsiToOem( const S : AnsiString) : AnsiString
3469:  Procedure StrAddRef( var S : AnsiString)
3470:  Function StrAllocSize( const S : AnsiString) : Longint
3471:  Procedure StrDecRef( var S : AnsiString)
3472:  //Function StrLen( S : PAnsiChar) : Integer
3473:  Function StrLength( const S : AnsiString) : Longint
3474:  Function StrRefCount( const S : AnsiString) : Longint
3475:  Procedure StrResetLength( var S : AnsiString)
3476:  Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3477:  Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3478:  Function StrStrCount( const S, SubS : AnsiString) : Integer
3479:  Function StrCompare( const S1, S2 : AnsiString) : Integer
3480:  Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3481:  //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3482:  Function StrFillChar1( const C : Char; Count : Integer) : AnsiString;
3483:  Function StrFillChar(const C: Char; Count: Integer): string)');
3484:  Function IntFillChar(const I: Integer; Count: Integer): string)');
3485:  Function ByteFillChar(const B: Byte; Count: Integer): string)');
3486:  Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;');
3487:  Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3488:  Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3489:  //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3490:  Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3491:  Function StrILastPos( const SubStr, S : AnsiString) : Integer
```

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

```
3578: Function ReplaceString( S : string; const OldPattern, NewPattern : string) : string
3579: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer)
3580: ******************************************uPSI_JvJvBDEUtils;***************
3581: //JvBDEUtils
3582: Function CreateDbLocate : TJvLocateObject
3583:  //Function CheckOpen( Status : DBIResult) : Boolean
3584: Procedure FetchAllRecords( DataSet : TBDEDataSet)
3585: Function TransActive( Database : TDatabase) : Boolean
3586: Function AsyncQrySupported( Database : TDatabase) : Boolean
3587: Function GetQuoteChar( Database : TDatabase) : string
3588: Procedure ExecuteQuery( const DbName, QueryText : string)
3589: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string)
3590: Function FieldLogicMap( FldType : TFieldType) : Integer
3591: Function FieldSubtypeMap( FldType : TFieldType) : Integer Value : string; Buffer : Pointer)
3592: Function GetAliasPath( const AliasName : string) : string
3593: Function IsDirectory( const DatabaseName : string) : Boolean
3594: Function GetBdeDirectory : string
3595: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) : Boolean
3596: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3597: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3598: Function DataSetRecNo( DataSet : TDataSet) : Longint
3599: Function DataSetRecordCount( DataSet : TDataSet) : Longint
3600: Function DataSetPositionStr( DataSet : TDataSet) : string
3601: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean)
3602: Function CurrentRecordDeleted( DataSet : TBDEDataSet) : Boolean
3603: Function IsFilterApplicable( DataSet : TDataSet) : Boolean
3604: Function IsBookmarkStable( DataSet : TBDEDataSet) : Boolean
3605: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)
3606: Procedure RestoreIndex( Table : TTable)
3607: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)
3608: Procedure PackTable( Table : TTable)
3609: Procedure ReindexTable( Table : TTable)
3610: Procedure BdeFlushBuffers
3611: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer) : Pointer
3612: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3613: Procedure DbNotSupported
3614: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3615: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3616: Procedure
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3617: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3618: ******************************************uPSI_JvDateUtil;
3619: function CurrentYear: Word;
3620: function IsLeapYear(AYear: Integer): Boolean;
3621: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3622: function FirstDayOfPrevMonth: TDateTime;
3623: function LastDayOfPrevMonth: TDateTime;
3624: function FirstDayOfNextMonth: TDateTime;
3625: function ExtractDay(ADate: TDateTime): Word;
3626: function ExtractMonth(ADate: TDateTime): Word;
3627: function ExtractYear(ADate: TDateTime): Word;
3628: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3629: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3630: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3631: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3632: function ValidDate(ADate: TDateTime): Boolean;
3633: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3634: function MonthsBetween(Date1, Date2: TDateTime): Double;
3635: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3636: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3637: function DaysBetween(Date1, Date2: TDateTime): Longint;
3638: { The same as previous but if Date2 < Date1 result = 0 }
3639: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3640: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3641: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3642: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3643: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3644: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3645: { String to date conversions }
3646: function GetDateOrder(const DateFormat: string): TDateOrder;
3647: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3648: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3649: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3650: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3651: function DefDateFormat(FourDigitYear: Boolean): string;
3652: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3653: ------------------------------------------------------------------------------
3654: ******************************** JvUtils;*****************************
3655: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3656: function GetWordOnPos(const S: string; const P: Integer): string;
3657: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3658: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3659: { SubStr returns substring from string, S, separated with Separator string}
3660: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3661: { SubStrEnd same to previous function but Index numerated from the end of string }
3662: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3663: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
```

```
3664: function SubWord(P: PChar; var P2: PChar): string;
3665: { NumberByWord returns the text representation of
3666:   the number, N, in normal russian language. Was typed from Monitor magazine }
3667: function NumberByWord(const N: Longint): string;
3668: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3669:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3670: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3671: { GetXYByPos is same to previous function, but returns X position in line too}
3672: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3673: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3674: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3675: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3676: function ConcatSep(const S, S2, Separator: string): string;
3677: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3678: function ConcatLeftSep(const S, S2, Separator: string): string;
3679: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3680: function MinimizeString(const S: string; const MaxLen: Integer): string;
3681: { Next 4 function for russian chars transliterating.
3682:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3683: procedure Dos2Win(var S: string);
3684: procedure Win2Dos(var S: string);
3685: function Dos2WinRes(const S: string): string;
3686: function Win2DosRes(const S: string): string;
3687: function Win2Koi(const S: string): string;
3688: { Spaces returns string consists on N space chars }
3689: function Spaces(const N: Integer): string;
3690: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3691: function AddSpaces(const S: string; const N: Integer): string;
3692: { function LastDate for russian users only } { returns date relative to current date: '' }
3693: function LastDate(const Dat: TDateTime): string;
3694: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3695: function CurrencyToStr(const Cur: currency): string;
3696: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3697: function Cmp(const S1, S2: string): Boolean;
3698: { StringCat add S2 string to S1 and returns this string }
3699: function StringCat(var S1: string; S2: string): string;
3700: { HasChar returns True, if Char, Ch, contains in string, S }
3701: function HasChar(const Ch: Char; const S: string): Boolean;
3702: function HasAnyChar(const Chars: string; const S: string): Boolean;
3703: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3704: function CountOfChar(const Ch: Char; const S: string): Integer;
3705: function DefStr(const S: string; Default: string): string;
3706: {**** files routines}
3707: { GetWinDir returns Windows folder name }
3708: function GetWinDir: TFileName;
3709: function GetSysDir: String;
3710: { GetTempDir returns Windows temporary folder name }
3711: function GetTempDir: string;
3712: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3713: function GenTempFileName(FileName: string): string;
3714: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3715: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3716: { ClearDir clears folder Dir }
3717: function ClearDir(const Dir: string): Boolean;
3718: { DeleteDir clears and than delete folder Dir }
3719: function DeleteDir(const Dir: string): Boolean;
3720: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3721: function FileEquMask(FileName, Mask: TFileName): Boolean;
3722: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3723:   Masks must be separated with comma (';') }
3724: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3725: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3726: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3727: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3728: { FileGetInfo fills SearchRec record for specified file attributes}
3729: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3730: { HasSubFolder returns True, if folder APath contains other folders }
3731: function HasSubFolder(APath: TFileName): Boolean;
3732: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3733: function IsEmptyFolder(APath: TFileName): Boolean;
3734: { AddSlash add slash Char to Dir parameter, if needed }
3735: procedure AddSlash(var Dir: TFileName);
3736: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3737: function AddSlash2(const Dir: TFileName): string;
3738: { AddPath returns FileName with Path, if FileName not contain any path }
3739: function AddPath(const FileName, Path: TFileName): TFileName;
3740: function AddPaths(const PathList, Path: string): string;
3741: function ParentPath(const Path: TFileName): TFileName;
3742: function FindInPath(const FileName, PathList: string): TFileName;
3743: function FindInPaths(const fileName,paths: String): String;
3744: {$IFNDEF BCB1}
3745: { BrowseForFolder displays Browse For Folder dialog }
3746: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3747: {$ENDIF BCB1}
3748: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3749: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3750: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
```

```
3751: function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3752:
3753: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3754: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3755: { HasParam returns True, if program running with specified parameter, Param }
3756: function HasParam(const Param: string): Boolean;
3757: function HasSwitch(const Param: string): Boolean;
3758: function Switch(const Param: string): string;
3759: { ExePath returns ExtractFilePath(ParamStr(0)) }
3760: function ExePath: TFileName;
3761: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3762: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3763: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3764: {**** Graphic routines }
3765: { TTFontSelected returns True, if True Type font is selected in specified device context }
3766: function TTFontSelected(const DC: HDC): Boolean;
3767: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3768: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3769: {**** Windows routines }
3770: { SetWindowTop put window to top without recreating window }
3771: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3772: {**** other routines }
3773: { KeyPressed returns True, if Key VK is now pressed }
3774: function KeyPressed(VK: Integer): Boolean;
3775: procedure SwapInt(var Int1, Int2: Integer);
3776: function IntPower(Base, Exponent: Integer): Integer;
3777: function ChangeTopException(E: TObject): TObject;
3778: function StrToBool(const S: string): Boolean;
3779: {$IFNDEF COMPILER3_UP}
3780: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3781:   Length of MaxLen bytes. The compare operation is controlled by the
3782:   current Windows locale. The return value is the same as for CompareStr. }
3783: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3784: function AnsiStrIComp(S1, S2: PChar): Integer;
3785: {$ENDIF}
3786: function Var2Type(V: Variant; const VarType: Integer): Variant;
3787: function VarToInt(V: Variant): Integer;
3788: function VarToFloat(V: Variant): Double;
3789: { following functions are not documented because they are don't work properly , so don't use them }
3790: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3791: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3792: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3793: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3794: function GetParameter: string;
3795: function GetLongFileName(FileName: string): string;
3796: {* from  FileCtrl}
3797: function DirectoryExists(const Name: string): Boolean;
3798: procedure ForceDirectories(Dir: string);
3799: {# from  FileCtrl}
3800: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3801: function GetComputerID: string;
3802: function GetComputerName: string;
3803: {**** string routines }
3804: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
      same Index.Also see RAUtilsW.ReplaceSokr1 function }
3805: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3806: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3807:   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 }
3808: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3809: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3810: function CountOfLines(const S: string): Integer;
3811: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3812: procedure DeleteEmptyLines(Ss: TStrings);
3813: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3814:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3815: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3816: {**** files routines - }
3817: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3818:   Resource can be compressed using MS Compress program}
3819: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3820: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3821: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3822: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3823: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3824: { IniReadSection read section, Section, from ini-file,
3825:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3826:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3827: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3828: { LoadTextFile load text file, FileName, into string }
3829: function LoadTextFile(const FileName: TFileName): string;
3830: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3831: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3832: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3833: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3834: {$IFDEF COMPILER3_UP}
3835: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3836: function TargetFileName(const FileName: TFileName): TFileName;
3837: { return filename ShortCut linked to }
```

```
3838: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3839: {$ENDIF COMPILER3_UP}
3840: {**** Graphic routines - }
3841: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3842: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3843: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3844: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3845: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3846: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3847: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3848: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3849: { Cinema draws some visual effect }
3850: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3851: { Roughed fills rect with special 3D pattern }
3852: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3853: { 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 }
3854: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3855: { TextWidth calculate text with for writing using standard desktop font }
3856: function TextWidth(AStr: string): Integer;
3857: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3858: function DefineCursor(Identifer: PChar): TCursor;
3859: {**** other routines - }
3860: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3861: function FindFormByClass(FormClass: TFormClass): TForm;
3862: function FindFormByClassName(FormClassName: string): TForm;
3863: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3864:   having Tag property value, equaled to Tag parameter }
3865: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3866: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3867: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3868: { RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3869: function RBTag(Parent: TWinControl): Integer;
3870: { AppMinimized returns True, if Application is minimized }
3871: function AppMinimized: Boolean;
3872: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3873:   if Caption parameter = '', it replaced with Application.Title }
3874: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
3875: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3876:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3877: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3878:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3879: { Delay stop program execution to MSec msec }
3880: procedure Delay(MSec: Longword);
3881: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3882: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3883: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3884: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3885: function PanelBorder(Panel: TCustomPanel): Integer;
3886: function Pixels(Control: TControl; APixels: Integer): Integer;
3887: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3888: procedure Error(const Msg: string);
3889: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3890:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3891:   {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3892: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3893:   const HideSelColor: Boolean): string;
3894: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3895:   const HideSelColor: Boolean): Integer;
3896: function ItemHtPlain(const Text: string): string;
3897: { ClearList - clears list of TObject }
3898: procedure ClearList(List: TList);
3899: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3900: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3901: { RTTI support }
3902: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3903: function GetPropStr(Obj: TObject; const PropName: string): string;
3904: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3905: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3906: procedure PrepareIniSection(SS: TStrings);
3907: { following functions are not documented because they are don't work properly, so don't use them }
3908: {$IFDEF COMPILER2}
3909: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3910: {$ENDIF}
3911:
3912: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3913: begin
3914:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3915:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3916:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
3917:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3918:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3919:  Procedure BoxSetItem( List : TWinControl; Index : Integer)
3920:  Function BoxGetFirstSelection( List : TWinControl) : Integer
3921:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3922: end;
3923:
3924: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
```

```
3925: begin
3926:  Const('MaxInitStrNum','LongInt'( 9);
3927: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
      : array of AnsiString; MaxSplit : Integer) : Integer
3928: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
      string; MaxSplit : Integer) : Integer
3929: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
      QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3930: Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3931: Function JvStrStrip( S : string) : string
3932: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3933: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
3934: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3935: Function StrEatWhiteSpace( const S : string) : string
3936: Function HexToAscii( const S : AnsiString) : AnsiString
3937: Function AsciiToHex( const S : AnsiString) : AnsiString
3938: Function StripQuotes( const S1 : AnsiString) : AnsiString
3939: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3940: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3941: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3942: Function HexPCharToInt( S1 : PAnsiChar) : Integer
3943: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
3944: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
3945: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
3946: Function JvValidIdentifier( S1 : String) : Boolean
3947: Function JvEndChar( X : AnsiChar) : Boolean
3948: Procedure JvGetToken( S1, S2 : PAnsiChar)
3949: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
3950: Function IsKeyword( S1 : PAnsiChar) : Boolean
3951: Function JvValidVarReference( S1 : PAnsiChar) : Boolean
3952: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
3953: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
3954: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
3955: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
3956: Function GetTokenCount : Integer
3957: Procedure ResetTokenCount
3958: end;
3959:
3960: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
3961: begin
3962:   SIRegister_TJvQueryParamsDialog(CL);
3963:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
3964: end;
3965:
3966: ******************************** JvStrUtil /  JvStrUtils;****************************
3967: function FindNotBlankCharPos(const S: string): Integer;
3968: function AnsiChangeCase(const S: string): string;
3969: function GetWordOnPos(const S: string; const P: Integer): string;
3970: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3971: function Cmp(const S1, S2: string): Boolean;
3972: { Spaces returns string consists on N space chars }
3973: function Spaces(const N: Integer): string;
3974: { HasChar returns True, if char, Ch, contains in string, S }
3975: function HasChar(const Ch: Char; const S: string): Boolean;
3976: function HasAnyChar(const Chars: string; const S: string): Boolean;
3977: { SubStr returns substring from string, S, separated with Separator string}
3978: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3979: { SubStrEnd same to previous function but Index numerated from the end of string }
3980: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3981: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
3982: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3983: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3984: { GetXYByPos is same to previous function, but returns X position in line too}
3985: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3986: { AddSlash returns string with added slash char to Dir parameter, if needed }
3987: function AddSlash2(const Dir: TFileName): string;
3988: { AddPath returns FileName with Path, if FileName not contain any path }
3989: function AddPath(const FileName, Path: TFileName): TFileName;
3990: { ExePath returns ExtractFilePath(ParamStr(0)) }
3991: function ExePath: TFileName;
3992: function LoadTextFile(const FileName: TFileName): string;
3993: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3994: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3995: function ConcatSep(const S, S2, Separator: string): string;
3996: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3997: function FileEquMask(FileName, Mask: TFileName): Boolean;
3998: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3999:   Masks must be separated with comma (';') }
4000: function FileEquMasks(FileName, Masks: TFileName): Boolean;
4001: function StringEndsWith(const Str, SubStr: string): Boolean;
4002: function ExtractFilePath2(const FileName: string): string;
4003: function StrToOem(const AnsiStr: string): string;
4004: { StrToOem translates a string from the Windows character set into the OEM character set. }
4005: function OemToAnsiStr(const OemStr: string): string;
4006: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4007: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4008: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4009: function ReplaceStr(const S, Srch, Replace: string): string;
4010: { Returns string with every occurrence of Srch string replaced with Replace string. }
```

```
4011: function DelSpace(const S: string): string;
4012: { DelSpace return a string with all white spaces removed. }
4013: function DelChars(const S: string; Chr: Char): string;
4014: { DelChars return a string with all Chr characters removed. }
4015: function DelBSpace(const S: string): string;
4016: { DelBSpace trims leading spaces from the given string. }
4017: function DelESpace(const S: string): string;
4018: { DelESpace trims trailing spaces from the given string. }
4019: function DelRSpace(const S: string): string;
4020: { DelRSpace trims leading and trailing spaces from the given string. }
4021: function DelSpace1(const S: string): string;
4022: { DelSpace1 return a string with all non-single white spaces removed. }
4023: function Tab2Space(const S: string; Numb: Byte): string;
4024: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4025: function NPos(const C: string; S: string; N: Integer): Integer;
4026: { NPos searches for a N-th position of substring C in a given string. }
4027: function MakeStr(C: Char; N: Integer): string;
4028: function MS(C: Char; N: Integer): string;
4029: { MakeStr return a string of length N filled with character C. }
4030: function AddChar(C: Char; const S: string; N: Integer): string;
4031: { AddChar return a string left-padded to length N with characters C. }
4032: function AddCharR(C: Char; const S: string; N: Integer): string;
4033: { AddCharR return a string right-padded to length N with characters C. }
4034: function LeftStr(const S: string; N: Integer): string;
4035: { LeftStr return a string right-padded to length N with blanks. }
4036: function RightStr(const S: string; N: Integer): string;
4037: { RightStr return a string left-padded to length N with blanks. }
4038: function CenterStr(const S: string; Len: Integer): string;
4039: { CenterStr centers the characters in the string based upon the Len specified. }
4040: function CompStr(const S1, S2: string): Integer;
4041: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4042: function CompText(const S1, S2: string): Integer;
4043: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4044: function Copy2Symb(const S: string; Symb: Char): string;
4045: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4046: function Copy2SymbDel(var S: string; Symb: Char): string;
4047: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4048: function Copy2Space(const S: string): string;
4049: { Copy2Symb returns a substring of a string S from begining to first white space. }
4050: function Copy2SpaceDel(var S: string): string;
4051: { Copy2SpaceDel returns a substring of a string S from begining to first
4052:   white space and removes this substring from S. }
4053: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4054: { Returns string, with the first letter of each word in uppercase,
4055:   all other letters in lowercase. Words are delimited by WordDelims. }
4056: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4057: { WordCount given a set of word delimiters, returns number of words in S. }
4058: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4059: { Given a set of word delimiters, returns start position of N'th word in S. }
4060: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4061: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4062: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4063: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4064:   delimiters, return the N'th word in S. }
4065: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4066: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4067: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4068: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4069: function QuotedString(const S: string; Quote: Char): string;
4070: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4071: function ExtractQuotedString(const S: string; Quote: Char): string;
4072: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4073:    and reduces pairs of Quote characters within the quoted string to a single character. }
4074: function FindPart(const HelpWilds, InputStr: string): Integer;
4075: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4076: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4077: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4078: function XorString(const Key, Src: ShortString): ShortString;
4079: function XorEncode(const Key, Source: string): string;
4080: function XorDecode(const Key, Source: string): string;
4081: { ** Command line routines ** }
4082: {$IFNDEF COMPILER4_UP}
4083: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4084: {$ENDIF}
4085: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4086: { ** Numeric string handling routines ** }
4087: function Numb2USA(const S: string): string;
4088: { Numb2USA converts numeric string S to USA-format. }
4089: function Dec2Hex(N: Longint; A: Byte): string;
4090: function D2H(N: Longint; A: Byte): string;
4091: { Dec2Hex converts the given value to a hexadecimal string representation
4092:   with the minimum number of digits (A) specified. }
4093: function Hex2Dec(const S: string): Longint;
4094: function H2D(const S: string): Longint;
4095: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4096: function Dec2Numb(N: Longint; A, B: Byte): string;
4097: { Dec2Numb converts the given value to a string representation with the
4098:   base equal to B and with the minimum number of digits (A) specified. }
4099: function Numb2Dec(S: string; B: Byte): Longint;
```

```
4100: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4101: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4102: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4103: function IntToRoman(Value: Longint): string;
4104: { IntToRoman converts the given value to a roman numeric string representation. }
4105: function RomanToInt(const S: string): Longint;
4106: { RomanToInt converts the given string to an integer value. If the string
4107:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4108: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4109: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4110: ********************************* JvFileUtil;*********************************
4111: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4112: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4113: procedure MoveFile(const FileName, DestName: TFileName);
4114: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4115: {$IFDEF COMPILER4_UP}
4116: function GetFileSize(const FileName: string): Int64;
4117: {$ELSE}
4118: function GetFileSize(const FileName: string): Longint;
4119: {$ENDIF}
4120: function FileDateTime(const FileName: string): TDateTime;
4121: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4122: function DeleteFiles(const FileMask: string): Boolean;
4123: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4124: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4125: function NormalDir(const DirName: string): string;
4126: function RemoveBackSlash(const DirName: string): string;
4127: function ValidFileName(const FileName: string): Boolean;
4128: function DirExists(Name: string): Boolean;
4129: procedure ForceDirectories(Dir: string);
4130: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4131:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4132: {$IFDEF COMPILER4_UP}
4133: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4134: {$ENDIF}
4135: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4136: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4137: {$IFDEF COMPILER4_UP}
4138: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4139: {$ENDIF}
4140: function GetTempDir: string;
4141: function GetWindowsDir: string;
4142: function GetSystemDir: string;
4143: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4144: {$IFDEF WIN32}
4145: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4146: function ShortToLongFileName(const ShortName: string): string;
4147: function ShortToLongPath(const ShortName: string): string;
4148: function LongToShortFileName(const LongName: string): string;
4149: function LongToShortPath(const LongName: string): string;
4150: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4151: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4152: {$ENDIF WIN32}
4153: {$IFNDEF COMPILER3_UP}
4154: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4155: {$ENDIF}
4156: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4157: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4158: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4159: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4160:
4161: //*************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4162:  Procedure VariantClear( var V : Variant)
4163:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4164:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4165:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4166:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4167:  Procedure VariantSub( const src : Variant; var dst : Variant)
4168:  Procedure VariantMul( const src : Variant; var dst : Variant)
4169:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4170:  Procedure VariantMod( const src : Variant; var dst : Variant)
4171:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4172:  Procedure VariantOr( const src : Variant; var dst : Variant)
4173:  Procedure VariantXor( const src : Variant; var dst : Variant)
4174:  Procedure VariantShl( const src : Variant; var dst : Variant)
4175:  Procedure VariantShr( const src : Variant; var dst : Variant)
4176:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4177:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4178:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4179:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4180:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4181:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4182:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4183:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4184:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4185:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4186:  Function VariantNot( const V1 : Variant) : Variant
4187:  Function VariantNeg( const V1 : Variant) : Variant
```

```
4188:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4189:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4190:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4191:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4192:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4193:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4194:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4195:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4196:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4197:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4198: end;
4199:
4200: ************************************unit uPSI_JvgUtils;************************************
4201: function IsEven(I: Integer): Boolean;
4202: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4203: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4204: procedure SwapInt(var I1, I2: Integer);
4205: function Spaces(Count: Integer): string;
4206: function DupStr(const Str: string; Count: Integer): string;
4207: function DupChar(C: Char; Count: Integer): string;
4208: procedure Msg(const AMsg: string);
4209: function RectW(R: TRect): Integer;
4210: function RectH(R: TRect): Integer;
4211: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4212: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4213: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4214: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4215:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4216: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4217: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4218:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor,HighlightColor,ShadowColor:
      TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4219: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4220: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4221:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4222: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4223: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4224: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4225:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4226:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4227:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4228: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4229:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4230:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4231:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4232: procedure BringParentWindowToTop(Wnd: TWinControl);
4233: function GetParentForm(Control: TControl): TForm;
4234: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4235: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4236: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4237: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4238: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4239: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4240: function CalcMathString(AExpression: string): Single;
4241: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4242: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4243: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4244: procedure TypeStringOnKeyboard(const S: string);
4245: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4246: procedure DrawTextExtAligned(Canvas:TCanvas;const
      Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4247: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4248: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4249: function ComponentToString(Component: TComponent): string;
4250: procedure StringToComponent(Component: TComponent; const Value: string);
4251: function PlayWaveResource(const ResName: string): Boolean;
4252: function UserName: string;
4253: function ComputerName: string;
4254: function CreateIniFileName: string;
4255: function ExpandString(const Str: string; Len: Integer): string;
4256: function Transliterate(const Str: string; RusToLat: Boolean): string;
4257: function IsSmallFonts: Boolean;
4258: function SystemColorDepth: Integer;
4259: function GetFileTypeJ(const FileName: string): TglFileType;
4260: Function GetFileType( hFile : THandle) : DWORD');
4261: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4262: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4263:
4264: { ************************************Utility routines of unit classes}
4265: function LineStart(Buffer, BufPos: PChar): PChar
4266: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4267:   'Strings: TStrings): Integer
4268:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4269:  Procedure RegisterClass( AClass : TPersistentClass)
4270:  Procedure RegisterClasses( AClasses : array of TPersistentClass)
4271:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4272:  Procedure UnRegisterClass( AClass : TPersistentClass)
4273:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4274:  Procedure UnRegisterModuleClasses( Module : HMODULE)
```

```
4275:  Function FindGlobalComponent( const Name : string) : TComponent
4276:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4277:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4278:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4279:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4280:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4281:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4282:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4283:  Procedure GlobalFixupReferences
4284:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4285:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4286:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4287:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4288:  Procedure RemoveFixups( Instance : TPersistent)
4289:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
4290:  Procedure BeginGlobalLoading
4291:  Procedure NotifyGlobalLoading
4292:  Procedure EndGlobalLoading
4293:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4294:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4295:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4296:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4297:  Procedure FreeObjectInstance( ObjectInstance : Pointer)
4298: // Function AllocateHWnd( Method : TWndMethod) : HWND
4299:  Procedure DeallocateHWnd( Wnd : HWND)
4300:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4301: ************************************unit uPSI_SqlTimSt and DB;*********************************
4302: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4303: Function VarSQLTimeStampCreate3: Variant;
4304: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4305: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4306: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4307: Function VarSQLTimeStamp : TVarType
4308: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4309: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4310: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4311: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4312: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4313: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4314: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4315: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4316: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4317: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4318: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4319: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4320: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4321:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4322: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4323: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4324: Procedure DisposeMem( var Buffer, Size : Integer)
4325: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4326: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4327: Function VarTypeToDataType( VarType : Integer) : TFieldType
4328: ******************************************unit JvStrings;*************************************
4329: {template functions}
4330: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4331: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4332: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4333: function RemoveMasterBlocks(const SourceStr: string): string;
4334: function RemoveFields(const SourceStr: string): string;
4335: {http functions}
4336: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4337: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4338: {set functions}
4339: procedure SplitSet(AText: string; AList: TStringList);
4340: function JoinSet(AList: TStringList): string;
4341: function FirstOfSet(const AText: string): string;
4342: function LastOfSet(const AText: string): string;
4343: function CountOfSet(const AText: string): Integer;
4344: function SetRotateRight(const AText: string): string;
4345: function SetRotateLeft(const AText: string): string;
4346: function SetPick(const AText: string; AIndex: Integer): string;
4347: function SetSort(const AText: string): string;
4348: function SetUnion(const Set1, Set2: string): string;
4349: function SetIntersect(const Set1, Set2: string): string;
4350: function SetExclude(const Set1, Set2: string): string;
4351: {replace any <,> etc by &lt; &gt;}
4352: function XMLSafe(const AText: string): string;
4353: {simple hash, Result can be used in Encrypt}
4354: function Hash(const AText: string): Integer;
4355: { Base64 encode and decode a string }
4356: function B64Encode(const S: AnsiString): AnsiString;
4357: function B64Decode(const S: AnsiString): AnsiString;
4358: {Basic encryption from a Borland Example}
4359: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4360: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4361: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4362: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4363: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
```

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

```
4451:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4452: // finds a text range, e.g. <TD>....</td> case insensitive
4453: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4454:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4455: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4456: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4457:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4458: // finds a text range backward, e.g. <TD>....</td> case insensitive
4459: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4460:   var RangeEnd: Integer): Boolean;
4461: // finds a HTML or XML tag:  <....>
4462: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4463:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4464: // finds the innertext between opening and closing tags
4465: function Easter(NYear: Integer): TDateTime;
4466: // returns the easter date of a year.
4467: function GetWeekNumber(Today: TDateTime): string;
4468: //gets a datecode. Returns year and weeknumber in format: YYWW
4469: function ParseNumber(const S: string): Integer;
4470: // parse number returns the last position, starting from 1
4471: function ParseDate(const S: string): Integer;
4472: // parse a SQL style data string from positions 1,
4473: // starts and ends with #
4474:
4475: ****************************************unit JvJCLUtils;****************************************
4476:
4477: function VarIsInt(Value: Variant): Boolean;
4478:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4479: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4480: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4481: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4482: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4483: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4484: function GetWordOnPos(const S: string; const P: Integer): string;
4485: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4486: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4487: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4488: { GetWordOnPosEx working like GetWordOnPos function, but
4489:   also returns Word position in iBeg, iEnd variables }
4490: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4491: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4492: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4493: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4494: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4495: { GetEndPosCaret returns the caret position of the last char. For the position
4496:   after the last char of Text you must add 1 to the returned X value. }
4497: procedure GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Integer;var X,Y:Integer);
4498: { GetEndPosCaret returns the caret position of the last char. For the position
4499:   after the last char of Text you must add 1 to the returned X value. }
4500: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4501: function SubStrBySeparator(const S:string;const Index:Integer;const
     Separator:string;StartIndex:Int=1):string;
4502: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
     Separator:WideString;StartIndex:Int:WideString;
4503: { SubStrEnd same to previous function but Index numerated from the end of string }
4504: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4505: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4506: function SubWord(P: PChar; var P2: PChar): string;
4507: function CurrencyByWord(Value: Currency): string;
4508: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4509: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4510: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4511: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4512: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4513: { ReplaceString searches for all substrings, OldPattern,
4514:   in a string, S, and replaces them with NewPattern }
4515: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4516: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
     WideString;StartIndex:Integer=1):WideString;
4517: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4518: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4519: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4520: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4521:
4522: { Next 4 function for russian chars transliterating.
4523:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4524: procedure Dos2Win(var S: AnsiString);
4525: procedure Win2Dos(var S: AnsiString);
4526: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4527: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4528: function Win2Koi(const S: AnsiString): AnsiString;
4529: { FillString fills the string Buffer with Count Chars }
4530: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4531: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4532: { MoveString copies Count Chars from Source to Dest }
4533: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
     inline; {$ENDIF SUPPORTS_INLINE} overload;
```

```
4534: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4535:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4536: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4537: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4538: { MoveWideChar copies Count WideChars from Source to Dest }
4539: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
      SUPPORTS_INLINE}
4540: { FillNativeChar fills Buffer with Count NativeChars }
4541: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4542: { MoveWideChar copies Count WideChars from Source to Dest }
4543: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4544: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4545: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4546: { Spaces returns string consists on N space chars }
4547: function Spaces(const N: Integer): string;
4548: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4549: function AddSpaces(const S: string; const N: Integer): string;
4550: function SpacesW(const N: Integer): WideString;
4551: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4552: { function LastDateRUS for russian users only }
4553: { returns date relative to current date: 'äâà äíÿ íàçàä' }
4554: function LastDateRUS(const Dat: TDateTime): string;
4555: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4556: function CurrencyToStr(const Cur: Currency): string;
4557: { HasChar returns True, if Char, Ch, contains in string, S }
4558: function HasChar(const Ch: Char; const S: string): Boolean;
4559: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4560: function HasAnyChar(const Chars: string; const S: string): Boolean;
4561: {$IFNDEF COMPILER12_UP}
4562: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4563: {$ENDIF ~COMPILER12_UP}
4564: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
      SUPPORTS_INLINE}
4565: function CountOfChar(const Ch: Char; const S: string): Integer;
4566: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4567: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4568: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4569: function StrPosW(S, SubStr: PWideChar): PWideChar;
4570: function StrLenW(S: PWideChar): Integer;
4571: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4572: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4573: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4574: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4575: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4576:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4577: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4578: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4579: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4580: Procedure GrayscaleBitmap( Bitmap : TBitmap)
4581: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream
4582: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4583: Function ScreenPixelFormat : TPixelFormat
4584: Function ScreenColorCount : Integer
4585: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4586: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4587: //  SIRegister_TJvGradient(CL);
4588:
4589: {************************************* files routines}
4590: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4591: const
4592:   {$IFDEF MSWINDOWS}
4593:   DefaultCaseSensitivity = False;
4594:   {$ENDIF MSWINDOWS}
4595:   {$IFDEF UNIX}
4596:   DefaultCaseSensitivity = True;
4597:   {$ENDIF UNIX}
4598: { GenTempFileName returns temporary file name on
4599:   drive, there FileName is placed }
4600: function GenTempFileName(FileName: string): string;
4601: { GenTempFileNameExt same to previous function, but
4602:   returning filename has given extension, FileExt }
4603: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4604: { ClearDir clears folder Dir }
4605: function ClearDir(const Dir: string): Boolean;
4606: { DeleteDir clears and than delete folder Dir }
4607: function DeleteDir(const Dir: string): Boolean;
4608: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4609: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4610: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4611:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4612: function FileEquMasks(FileName, Masks: TFileName;
4613:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4614: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4615: {$IFDEF MSWINDOWS}
4616: { LZFileExpand expand file, FileSource,
```

```
4617:   into FileDest. Given file must be compressed, using MS Compress program }
4618: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4619: {$ENDIF MSWINDOWS}
4620: { FileGetInfo fills SearchRec record for specified file attributes}
4621: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4622: { HasSubFolder returns True, if folder APath contains other folders }
4623: function HasSubFolder(APath: TFileName): Boolean;
4624: { IsEmptyFolder returns True, if there are no files or
4625:   folders in given folder, APath}
4626: function IsEmptyFolder(APath: TFileName): Boolean;
4627: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4628: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4629: { AddPath returns FileName with Path, if FileName not contain any path }
4630: function AddPath(const FileName, Path: TFileName): TFileName;
4631: function AddPaths(const PathList, Path: string): string;
4632: function ParentPath(const Path: TFileName): TFileName;
4633: function FindInPath(const FileName, PathList: string): TFileName;
4634: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4635: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4636: { HasParam returns True, if program running with specified parameter, Param }
4637: function HasParam(const Param: string): Boolean;
4638: function HasSwitch(const Param: string): Boolean;
4639: function Switch(const Param: string): string;
4640: { ExePath returns ExtractFilePath(ParamStr(0)) }
4641: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4642: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4643: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4644: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4645: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4646: {**** Graphic routines }
4647: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4648: function IsTTFontSelected(const DC: HDC): Boolean;
4649: function KeyPressed(VK: Integer): Boolean;
4650: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4651: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4652: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4653: {**** Color routines }
4654: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4655: function RGBToBGR(Value: Cardinal): Cardinal;
4656: //function ColorToPrettyName(Value: TColor): string;
4657: //function PrettyNameToColor(const Value: string): TColor;
4658: {**** other routines }
4659: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4660: function IntPower(Base, Exponent: Integer): Integer;
4661: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4662: function StrToBool(const S: string): Boolean;
4663: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4664: function VarToInt(V: Variant): Integer;
4665: function VarToFloat(V: Variant): Double;
4666: { following functions are not documented because they not work properly sometimes, so do not use them }
4667: // (rom) ReplaceStrings1, GetSubStr removed
4668: function GetLongFileName(const FileName: string): string;
4669: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4670: function GetParameter: string;
4671: function GetComputerID: string;
4672: function GetComputerName: string;
4673: {**** string routines }
4674: { ReplaceAllStrings searches for all substrings, Words,
4675:   in a string, S, and replaces them with Frases with the same Index. }
4676: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4677: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4678:   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 }
4679: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4680: { CountOfLines calculates the lines count in a string, S,
4681:   each line must be separated from another with CrLf sequence }
4682: function CountOfLines(const S: string): Integer;
4683: { DeleteLines deletes all lines from strings which in the words,  words.
4684:   The word of will be deleted from strings. }
4685: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4686: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4687:   Lines contained only spaces also deletes. }
4688: procedure DeleteEmptyLines(Ss: TStrings);
4689: { SQLAddWhere addes or modifies existing where-statement, where,
4690:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4691:   it must be started on the begining of any line }
4692: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4693: {**** files routines - }
4694: {$IFDEF MSWINDOWS}
4695: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4696:   Resource can be compressed using MS Compress program}
4697: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4698: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
     Boolean;
4699: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4700: {$ENDIF MSWINDOWS}
4701: { IniReadSection read section, Section, from ini-file,
4702:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4703:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
```

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

```
4791: function DaysBetween(Date1, Date2: TDateTime): Longint;
4792: { The same as previous but if Date2 < Date1 result = 0 }
4793: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4794: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4795: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4796: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4797: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4798: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4799: { String to date conversions }
4800: function GetDateOrder(const DateFormat: string): TDateOrder;
4801: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4802: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4803: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4804: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4805: //function DefDateFormat(AFourDigitYear: Boolean): string;
4806: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4807: function FormatLongDate(Value: TDateTime): string;
4808: function FormatLongDateTime(Value: TDateTime): string;
4809: { end JvDateUtil }
4810: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4811: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4812: { begin JvStrUtils } { ** Common string handling routines ** }
4813: {$IFDEF UNIX}
4814: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4815:   const ToCode, FromCode: AnsiString): Boolean;
4816: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4817: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4818: function OemStrToAnsi(const S: AnsiString): AnsiString;
4819: function AnsiStrToOem(const S: AnsiString): AnsiString;
4820: {$ENDIF UNIX}
4821: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4822: { StrToOem translates a string from the Windows character set into the OEM character set. }
4823: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4824: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4825: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4826: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4827: function ReplaceStr(const S, Srch, Replace: string): string;
4828: { Returns string with every occurrence of Srch string replaced with Replace string. }
4829: function DelSpace(const S: string): string;
4830: { DelSpace return a string with all white spaces removed. }
4831: function DelChars(const S: string; Chr: Char): string;
4832: { DelChars return a string with all Chr characters removed. }
4833: function DelBSpace(const S: string): string;
4834: { DelBSpace trims leading spaces from the given string. }
4835: function DelESpace(const S: string): string;
4836: { DelESpace trims trailing spaces from the given string. }
4837: function DelRSpace(const S: string): string;
4838: { DelRSpace trims leading and trailing spaces from the given string. }
4839: function DelSpace1(const S: string): string;
4840: { DelSpace1 return a string with all non-single white spaces removed. }
4841: function Tab2Space(const S: string; Numb: Byte): string;
4842: { Tab2Space converts any tabulation character in the given string to the
4843:   Numb spaces characters. }
4844: function NPos(const C: string; S: string; N: Integer): Integer;
4845: { NPos searches for a N-th position of substring C in a given string. }
4846: function MakeStr(C: Char; N: Integer): string; overload;
4847: {$IFNDEF COMPILER12_UP}
4848: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4849: {$ENDIF !COMPILER12_UP}
4850: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4851: { MakeStr return a string of length N filled with character C. }
4852: function AddChar(C: Char; const S: string; N: Integer): string;
4853: { AddChar return a string left-padded to length N with characters C. }
4854: function AddCharR(C: Char; const S: string; N: Integer): string;
4855: { AddCharR return a string right-padded to length N with characters C. }
4856: function LeftStr(const S: string; N: Integer): string;
4857: { LeftStr return a string right-padded to length N with blanks. }
4858: function RightStr(const S: string; N: Integer): string;
4859: { RightStr return a string left-padded to length N with blanks. }
4860: function CenterStr(const S: string; Len: Integer): string;
4861: { CenterStr centers the characters in the string based upon the Len specified. }
4862: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4863: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4864:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4865: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4866: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4867: function Copy2Symb(const S: string; Symb: Char): string;
4868: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4869: function Copy2SymbDel(var S: string; Symb: Char): string;
4870: { Copy2SymbDel returns a substring of a string S from begining to first
4871:   character Symb and removes this substring from S. }
4872: function Copy2Space(const S: string): string;
4873: { Copy2Symb returns a substring of a string S from begining to first white space. }
4874: function Copy2SpaceDel(var S: string): string;
4875: { Copy2SpaceDel returns a substring of a string S from begining to first
4876:   white space and removes this substring from S. }
4877: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4878: { Returns string, with the first letter of each word in uppercase,
4879:   all other letters in lowercase. Words are delimited by WordDelims. }
```

```
4880: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4881: { WordCount given a set of word delimiters, returns number of words in S. }
4882: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4883: { Given a set of word delimiters, returns start position of N'th word in S. }
4884: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4885: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4886: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4887: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4888:   delimiters, return the N'th word in S. }
4889: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4890: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4891:   that started from position Pos. }
4892: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4893: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4894: function QuotedString(const S: string; Quote: Char): string;
4895: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4896: function ExtractQuotedString(const S: string; Quote: Char): string;
4897: { ExtractQuotedString removes the Quote characters from the beginning and
4898:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4899: function FindPart(const HelpWilds, InputStr: string): Integer;
4900: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4901: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4902: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4903: function XorString(const Key, Src: ShortString): ShortString;
4904: function XorEncode(const Key, Source: string): string;
4905: function XorDecode(const Key, Source: string): string;
4906: { ** Command line routines ** }
4907: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4908: { ** Numeric string handling routines ** }
4909: function Numb2USA(const S: string): string;
4910: { Numb2USA converts numeric string S to USA-format. }
4911: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4912: { Dec2Hex converts the given value to a hexadecimal string representation
4913:   with the minimum number of digits (A) specified. }
4914: function Hex2Dec(const S: string): Longint;
4915: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4916: function Dec2Numb(N: Int64; A, B: Byte): string;
4917: { Dec2Numb converts the given value to a string representation with the
4918:   base equal to B and with the minimum number of digits (A) specified. }
4919: function Numb2Dec(S: string; B: Byte): Int64;
4920: { Numb2Dec converts the given B-based numeric string to the corresponding
4921:   integer value. }
4922: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4923: { IntToBin converts the given value to a binary string representation
4924:   with the minimum number of digits specified. }
4925: function IntToRoman(Value: Longint): string;
4926: { IntToRoman converts the given value to a roman numeric string representation. }
4927: function RomanToInt(const S: string): Longint;
4928: { RomanToInt converts the given string to an integer value. If the string
4929:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4930: function FindNotBlankCharPos(const S: string): Integer;
4931: function FindNotBlankCharPosW(const S: WideString): Integer;
4932: function AnsiChangeCase(const S: string): string;
4933: function WideChangeCase(const S: string): string;
4934: function StartsText(const SubStr, S: string): Boolean;
4935: function EndsText(const SubStr, S: string): Boolean;
4936: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4937: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4938: {end JvStrUtils}
4939: {$IFDEF UNIX}
4940: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4941: {$ENDIF UNIX}
4942: { begin JvFileUtil }
4943: function FileDateTime(const FileName: string): TDateTime;
4944: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4945: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4946: function NormalDir(const DirName: string): string;
4947: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4948: function ValidFileName(const FileName: string): Boolean;
4949: {$IFDEF MSWINDOWS}
4950: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4951: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4952: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4953: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4954: {$ENDIF MSWINDOWS}
4955: function GetWindowsDir: string;
4956: function GetSystemDir: string;
4957: function ShortToLongFileName(const ShortName: string): string;
4958: function LongToShortFileName(const LongName: string): string;
4959: function ShortToLongPath(const ShortName: string): string;
4960: function LongToShortPath(const LongName: string): string;
4961: {$IFDEF MSWINDOWS}
4962: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4963: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4964: {$ENDIF MSWINDOWS}
4965: { end JvFileUtil }
4966: // Works like PtInRect but includes all edges in comparision
4967: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4968: // Works like PtInRect but excludes all edges from comparision
```

```
4969: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4970: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4971: function IsFourDigitYear: Boolean;
4972: { moved from JvJVCLUTils }
4973: //Open an object with the shell (url or something like that)
4974: function OpenObject(const Value: string): Boolean; overload;
4975: function OpenObject(Value: PChar): Boolean; overload;
4976: {$IFDEF MSWINDOWS}
4977: //Raise the last Exception
4978: procedure RaiseLastWin32; overload;
4979: procedure RaiseLastWin32(const Text: string); overload;
4980: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
4981: function GetFileVersion(const AFileName: string): Cardinal;
4982: {$EXTERNALSYM GetFileVersion}
4983: //Get version of Shell.dll
4984: function GetShellVersion: Cardinal;
4985: {$EXTERNALSYM GetShellVersion}
4986: // CD functions  on HW
4987: procedure OpenCdDrive;
4988: procedure CloseCdDrive;
4989: // returns True if Drive is accessible
4990: function DiskInDrive(Drive: Char): Boolean;
4991: {$ENDIF MSWINDOWS}
4992: //Same as linux function ;)
4993: procedure PError(const Text: string);
4994: // execute a program without waiting
4995: procedure Exec(const FileName, Parameters, Directory: string);
4996: // execute a program and wait for it to finish
4997: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
4998: // returns True if this is the first instance of the program that is running
4999: function FirstInstance(const ATitle: string): Boolean;
5000: // restores a window based on it's classname and Caption. Either can be left empty
5001: // to widen the search
5002: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5003: // manipulate the traybar and start button
5004: procedure HideTraybar;
5005: procedure ShowTraybar;
5006: procedure ShowStartButton(Visible: Boolean = True);
5007: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5008: procedure MonitorOn;
5009: procedure MonitorOff;
5010: procedure LowPower;
5011: // send a key to the window named AppName
5012: function SendKey(const AppName: string; Key: Char): Boolean;
5013: {$IFDEF MSWINDOWS}
5014: // returns a list of all win currently visible, the Objects property is filled with their window handle
5015: procedure GetVisibleWindows(List: TStrings);
5016: Function GetVisibleWindowsF( List : TStrings):TStrings');
5017: // associates an extension to a specific program
5018: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5019: procedure AddToRecentDocs(const FileName: string);
5020: function GetRecentDocs: TStringList;
5021: {$ENDIF MSWINDOWS}
5022: function CharIsMoney(const Ch: Char): Boolean;
5023: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5024: function IntToExtended(I: Integer): Extended;
5025: { GetChangedText works out the new text given the current cursor pos & the key pressed
5026:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5027: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5028: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5029: //function StrIsInteger(const S: string): Boolean;
5030: function StrIsFloatMoney(const Ps: string): Boolean;
5031: function StrIsDateTime(const Ps: string): Boolean;
5032: function PreformatDateString(Ps: string): string;
5033: function BooleanToInteger(const B: Boolean): Integer;
5034: function StringToBoolean(const Ps: string): Boolean;
5035: function SafeStrToDateTime(const Ps: string): TDateTime;
5036: function SafeStrToDate(const Ps: string): TDateTime;
5037: function SafeStrToTime(const Ps: string): TDateTime;
5038: function StrDelete(const psSub, psMain: string): string;
5039:   { returns the fractional value of pcValue}
5040: function TimeOnly(pcValue: TDateTime): TTime;
5041: { returns the integral value of pcValue }
5042: function DateOnly(pcValue: TDateTime): TDate;
5043: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5044: const { TDateTime value used to signify Null value}
5045:   NullEquivalentDate: TDateTime = 0.0;
5046: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5047: // Replacement for Win32Check to avoid platform specific warnings in D6
5048: function OSCheck(RetVal: Boolean): Boolean;
5049: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5050:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5051:   not be forced to use FileCtrl unnecessarily }
5052: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5053: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5054: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5055: function MinimizeString(const S: string; const MaxLen: Integer): string;
```

```
5056: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
      SW_SHOWDEFAULT);
5057: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion function and calls it, returning major and
      minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't be
      found.}
5058: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5059: {$ENDIF MSWINDOWS}
5060: procedure ResourceNotFound(ResID: PChar);
5061: function EmptyRect: TRect;
5062: function RectWidth(R: TRect): Integer;
5063: function RectHeight(R: TRect): Integer;
5064: function CompareRect(const R1, R2: TRect): Boolean;
5065: procedure RectNormalize(var R: TRect);
5066: function RectIsSquare(const R: TRect): Boolean;
5067: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5068: //If AMaxSize = -1 ,then auto calc Square's max size
5069: {$IFDEF MSWINDOWS}
5070: procedure FreeUnusedOle;
5071: function GetWindowsVersion: string;
5072: function LoadDLL(const LibName: string): THandle;
5073: function RegisterServer(const ModuleName: string): Boolean;
5074: function UnregisterServer(const ModuleName: string): Boolean;
5075: {$ENDIF MSWINDOWS}
5076: { String routines }
5077: function GetEnvVar(const VarName: string): string;
5078: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5079: function StringToPChar(var S: string): PChar;
5080: function StrPAlloc(const S: string): PChar;
5081: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5082: function DropT(const S: string): string;
5083: { Memory routines }
5084: function AllocMemo(Size: Longint): Pointer;
5085: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5086: procedure FreeMemo(var fpBlock: Pointer);
5087: function GetMemoSize(fpBlock: Pointer): Longint;
5088: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5089: { Manipulate huge pointers routines }
5090: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5091: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5092: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5093: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5094: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5095: function WindowClassName(Wnd: THandle): string;
5096: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5097: procedure ActivateWindow(Wnd: THandle);
5098: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5099: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5100: { SetWindowTop put window to top without recreating window }
5101: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5102: procedure CenterWindow(Wnd: THandle);
5103: function MakeVariant(const Values: array of Variant): Variant;
5104: { Convert dialog units to pixels and backwards }
5105: {$IFDEF MSWINDOWS}
5106: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5107: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5108: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5109: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5110: {$ENDIF MSWINDOWS}
5111: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5112: {$IFDEF BCB}
5113: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5114: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5115: {$ELSE}
5116: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5117: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5118: {$ENDIF BCB}
5119: {$IFDEF MSWINDOWS}
5120: { BrowseForFolderNative displays Browse For Folder dialog }
5121: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5122: {$ENDIF MSWINDOWS}
5123: procedure AntiAlias(Clip: TBitmap);
5124: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5125: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5126:   ABitmap: TBitmap; const SourceRect: TRect);
5127: function IsTrueType(const FontName: string): Boolean;
5128: // Removes all non-numeric characters from AValue and returns the resulting string
5129: function TextToValText(const AValue: string): string;
5130: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5131: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5132: Function ReplaceRegExpr(const ARegExpr,AInputStr,
      AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5133: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5134: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5135:
5136: ****************************************************unit uPSI_JvTFUtils;
5137:  Function JExtractYear( ADate : TDateTime) : Word
5138:  Function JExtractMonth( ADate : TDateTime) : Word
5139:  Function JExtractDay( ADate : TDateTime) : Word
5140:  Function ExtractHours( ATime : TDateTime) : Word
```

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

```
5228:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5229:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5230:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5231:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5232:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5233: end;
5234:
5235: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5236: begin
5237:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5238:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5239:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5240:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5241:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5242:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5243:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5244:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5245:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5246:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5247:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5248:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5249:  'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5250:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5251:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5252:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5253:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5254:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5255:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5256:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5257:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5258:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5259:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5260:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5261:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5262:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5263:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5264:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5265:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5266:  'ESBLn10','Extended').setExtended( 2.3025850929940456840 2);
5267:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5268:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5269:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5270:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5271:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5272:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5273:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5274:  'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5275:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5276:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5277:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5278:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5279:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5280:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5281:  'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5282:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5283:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5284:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5285:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5286:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5287:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5288:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5289:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5290:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5291:   //LongWord', 'Cardinal
5292:   TBitList', 'Word
5293:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5294:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5295:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5296:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5297:  Function SameFloat( const X1, X2 : Extended) : Boolean
5298:  Function FloatIsZero( const X : Extended) : Boolean
5299:  Function FloatIsPositive( const X : Extended) : Boolean
5300:  Function FloatIsNegative( const X : Extended) : Boolean
5301:  Procedure IncLim( var B : Byte; const Limit : Byte)
5302:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5303:  Procedure IncLimW( var B : Word; const Limit : Word)
5304:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5305:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5306:  Procedure DecLim( var B : Byte; const Limit : Byte)
5307:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5308:  Procedure DecLimW( var B : Word; const Limit : Word)
5309:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5310:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5311:  Function MaxB( const B1, B2 : Byte) : Byte
5312:  Function MinB( const B1, B2 : Byte) : Byte
5313:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5314:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5315:  Function MaxW( const B1, B2 : Word) : Word
5316:  Function MinW( const B1, B2 : Word) : Word
```

```
5317:  Function esbMaxI( const B1, B2 : Integer) : Integer
5318:  Function esbMinI( const B1, B2 : Integer) : Integer
5319:  Function MaxL( const B1, B2 : LongInt) : LongInt
5320:  Function MinL( const B1, B2 : LongInt) : LongInt
5321:  Procedure SwapB( var B1, B2 : Byte)
5322:  Procedure SwapSI( var B1, B2 : ShortInt)
5323:  Procedure SwapW( var B1, B2 : Word)
5324:  Procedure SwapI( var B1, B2 : SmallInt)
5325:  Procedure SwapL( var B1, B2 : LongInt)
5326:  Procedure SwapI32( var B1, B2 : Integer)
5327:  Procedure SwapC( var B1, B2 : LongWord)
5328:  Procedure SwapInt64( var X, Y : Int64)
5329:  Function esbSign( const B : LongInt) : ShortInt
5330:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5331:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5332:  Function Max3Word( const X1, X2, X3 : Word) : Word
5333:  Function Min3Word( const X1, X2, X3 : Word) : Word
5334:  Function MaxBArray( const B : array of Byte) : Byte
5335:  Function MaxWArray( const B : array of Word) : Word
5336:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5337:  Function MaxIArray( const B : array of Integer) : Integer
5338:  Function MaxLArray( const B : array of LongInt) : LongInt
5339:  Function MinBArray( const B : array of Byte) : Byte
5340:  Function MinWArray( const B : array of Word) : Word
5341:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5342:  Function MinIArray( const B : array of Integer) : Integer
5343:  Function MinLArray( const B : array of LongInt) : LongInt
5344:  Function SumBArray( const B : array of Byte) : Byte
5345:  Function SumBArray2( const B : array of Byte) : Word
5346:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5347:  Function SumSIArray2( const B : array of ShortInt) : Integer
5348:  Function SumWArray( const B : array of Word) : Word
5349:  Function SumWArray2( const B : array of Word) : LongInt
5350:  Function SumIArray( const B : array of Integer) : Integer
5351:  Function SumLArray( const B : array of LongInt) : LongInt
5352:  Function SumLWArray( const B : array of LongWord) : LongWord
5353:  Function ESBDigits( const X : LongWord) : Byte
5354:  Function BitsHighest( const X : LongWord) : Integer
5355:  Function ESBBitsNeeded( const X : LongWord) : Integer
5356:  Function esbGCD( const X, Y : LongWord) : LongWord
5357:  Function esbLCM( const X, Y : LongInt) : Int64
5358:  //Function esbLCM( const X, Y : LongInt) : LongInt
5359:  Function RelativePrime( const X, Y : LongWord) : Boolean
5360:  Function Get87ControlWord : TBitList
5361:  Procedure Set87ControlWord( const CWord : TBitList)
5362:  Procedure SwapExt( var X, Y : Extended)
5363:  Procedure SwapDbl( var X, Y : Double)
5364:  Procedure SwapSing( var X, Y : Single)
5365:  Function esbSgn( const X : Extended) : ShortInt
5366:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5367:  Function ExtMod( const X, Y : Extended) : Extended
5368:  Function ExtRem( const X, Y : Extended) : Extended
5369:  Function CompMOD( const X, Y : Comp) : Comp
5370:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5371:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5372:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5373:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5374:  Function MaxExt( const X, Y : Extended) : Extended
5375:  Function MinExt( const X, Y : Extended) : Extended
5376:  Function MaxEArray( const B : array of Extended) : Extended
5377:  Function MinEArray( const B : array of Extended) : Extended
5378:  Function MaxSArray( const B : array of Single) : Single
5379:  Function MinSArray( const B : array of Single) : Single
5380:  Function MaxCArray( const B : array of Comp) : Comp
5381:  Function MinCArray( const B : array of Comp) : Comp
5382:  Function SumSArray( const B : array of Single) : Single
5383:  Function SumEArray( const B : array of Extended) : Extended
5384:  Function SumSqEArray( const B : array of Extended) : Extended
5385:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5386:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5387:  Function SumCArray( const B : array of Comp) : Comp
5388:  Function FactorialX( A : LongWord) : Extended
5389:  Function PermutationX( N, R : LongWord) : Extended
5390:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5391:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5392:  Function esbGeometricMean( const X : array of Extended) : Extended
5393:  Function esbHarmonicMean( const X : array of Extended) : Extended
5394:  Function ESBMean( const X : array of Extended) : Extended
5395:  Function esbSampleVariance( const X : array of Extended) : Extended
5396:  Function esbPopulationVariance( const X : array of Extended) : Extended
5397:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5398:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5399:  Function GetMedian( const SortedX : array of Extended) : Extended
5400:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5401:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5402:  Function ESBMagnitude( const X : Extended) : Integer
5403:  Function ESBTan( Angle : Extended) : Extended
5404:  Function ESBCot( Angle : Extended) : Extended
5405:  Function ESBCosec( const Angle : Extended) : Extended
```

```
5406:  Function ESBSec( const Angle : Extended) : Extended
5407:  Function ESBArcTan( X, Y : Extended) : Extended
5408:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5409:  Function ESBArcCos( const X : Extended) : Extended
5410:  Function ESBArcSin( const X : Extended) : Extended
5411:  Function ESBArcSec( const X : Extended) : Extended
5412:  Function ESBArcCosec( const X : Extended) : Extended
5413:  Function ESBLog10( const X : Extended) : Extended
5414:  Function ESBLog2( const X : Extended) : Extended
5415:  Function ESBLogBase( const X, Base : Extended) : Extended
5416:  Function Pow2( const X : Extended) : Extended
5417:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5418:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5419:  Function XtoY( const X, Y : Extended) : Extended
5420:  Function esbTenToY( const Y : Extended) : Extended
5421:  Function esbTwoToY( const Y : Extended) : Extended
5422:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5423:  Function esbISqrt( const I : LongWord) : Longword
5424:  Function ILog2( const I : LongWord) : LongWord
5425:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5426:  Function ESBArCosh( X : Extended) : Extended
5427:  Function ESBArSinh( X : Extended) : Extended
5428:  Function ESBArTanh( X : Extended) : Extended
5429:  Function ESBCosh( X : Extended) : Extended
5430:  Function ESBSinh( X : Extended) : Extended
5431:  Function ESBTanh( X : Extended) : Extended
5432:  Function InverseGamma( const X : Extended) : Extended
5433:  Function esbGamma( const X : Extended) : Extended
5434:  Function esbLnGamma( const X : Extended) : Extended
5435:  Function esbBeta( const X, Y : Extended) : Extended
5436:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5437: end;
5438:
5439: ******************************Integer Huge Cardinal Utils*************************
5440: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5441: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5442: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5443: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5444: Function BitCount_8( Value : byte) : integer
5445: Function BitCount_16( Value : uint16) : integer
5446: Function BitCount_32( Value : uint32) : integer
5447: Function BitCount_64( Value : uint64) : integer
5448: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5449:  Procedure ( CountPrimalityTests : integer)
5450:  Function gcd( a, b : THugeCardinal) : THugeCardinal
5451:  Function lcm( a, b : THugeCardinal) : THugeCardinal
5452:  Function isCoPrime( a, b : THugeCardinal) : boolean
5453:  Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5454:  Function hasSmallFactor( p : THugeCardinal) : boolean
5455:  //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
      NumbersTested: integer) : boolean
5456:  Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5457:  Const('StandardExponent','LongInt'( 65537);
5458: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5459:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5460:
5461: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5462: begin
5463:   AddTypeS('TXRTLInteger', 'array of Integer
5464:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5465:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5466:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5467:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5468:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5469:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5470:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5471:   'BitsPerByte','LongInt'( 8);
5472:   BitsPerDigit','LongInt'( 32);
5473:   SignBitMask','LongWord( $80000000);
5474:  Function XRTLAdjustBits( const ABits : Integer) : Integer
5475:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5476:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5477:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5478:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5479:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5480:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5481:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5482:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5483:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5484:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5485:  Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5486:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5487:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5488:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5489:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5490:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer
```

```
5491:  Procedure XRTLZero( var AInteger : TXRTLInteger)
5492:  Procedure XRTLOne( var AInteger : TXRTLInteger)
5493:  Procedure XRTLMOne( var AInteger : TXRTLInteger)
5494:  Procedure XRTLTwo( var AInteger : TXRTLInteger)
5495:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5496:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5497:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5498:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5499:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5500:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5501:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5502:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5503:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5504:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5505:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5506:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5507:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5508:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5509:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5510:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5511:  Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger)
5512:  Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5513:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5514:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5515:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5516:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5517:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5518:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5519:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5520:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5521:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5522:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5523:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5524:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5525:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5526:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5527:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5528:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5529:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5530:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5531:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5532:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5533:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5534:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5535:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5536:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5537:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5538:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5539:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5540:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5541:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5542:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5543:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5544:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5545:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5546:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5547:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5548: end;
5549:
5550: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5551: begin
5552:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5553:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5554:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5555:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5556:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const
      AColor:TColor;Rect:TRect;
5557:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5558:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5559:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5560:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5561:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5562:  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)
5563: end;
5564:
5565:
5566: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5567: begin
5568:  Function StrDec( S : String) : String
5569:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5570:  Function ReadNumFromEdit( Edit : TEdit) : Float
```

```
5571:  Procedure WriteNumToFile( var F : Text; X : Float)
5572: end;
5573:
5574: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5575: begin
5576:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5577:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5578:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5579:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5580:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5581:  Procedure TeX_SetGraphTitle( Title : String)
5582:  Procedure TeX_SetOxTitle( Title : String)
5583:  Procedure TeX_SetOyTitle( Title : String)
5584:  Procedure TeX_PlotOxAxis
5585:  Procedure TeX_PlotOyAxis
5586:  Procedure TeX_PlotGrid( Grid : TGrid)
5587:  Procedure TeX_WriteGraphTitle
5588:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5589:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5590:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5591:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5592:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5593:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5594:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5595:  Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5596:  Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5597:  Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5598:  Function Xcm( X : Float) : Float
5599:  Function Ycm( Y : Float) : Float
5600: end;
5601:
5602: *---------------------------------------------------------------------------*)
5603: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5604: begin
5605:   TConstArray', 'array of TVarRec
5606:  Function CopyVarRec( const Item : TVarRec) : TVarRec
5607:  Function CreateConstArray( const Elements : array of const) : TConstArray
5608:  Procedure FinalizeVarRec( var Item : TVarRec)
5609:  Procedure FinalizeConstArray( var Arr : TConstArray)
5610: end;
5611:
5612: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5613: begin
5614:  Function HexBS( B : Byte) : ShortString
5615:  Function HexWS( W : Word) : ShortString
5616:  Function HexLS( L : LongInt) : ShortString
5617:  Function HexPtrS( P : Pointer) : ShortString
5618:  Function BinaryBS( B : Byte) : ShortString
5619:  Function BinaryWS( W : Word) : ShortString
5620:  Function BinaryLS( L : LongInt) : ShortString
5621:  Function OctalBS( B : Byte) : ShortString
5622:  Function OctalWS( W : Word) : ShortString
5623:  Function OctalLS( L : LongInt) : ShortString
5624:  Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5625:  Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5626:  Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5627:  Function Str2RealS( const S : ShortString; var R : Double) : Boolean
5628:  Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5629:  Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5630:  Function Long2StrS( L : LongInt) : ShortString
5631:  Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5632:  Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5633:  Function ValPrepS( const S : ShortString) : ShortString
5634:  Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5635:  Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5636:  Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5637:  Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5638:  Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5639:  Function TrimLeadS( const S : ShortString) : ShortString
5640:  Function TrimTrailS( const S : ShortString) : ShortString
5641:  Function TrimS( const S : ShortString) : ShortString
5642:  Function TrimSpacesS( const S : ShortString) : ShortString
5643:  Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5644:  Function CenterS( const S : ShortString; Len : Cardinal) : ShortString
5645:  Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5646:  Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5647:  Function ScrambleS( const S, Key : ShortString) : ShortString
5648:  Function SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5649:  Function FilterS( const S, Filters : ShortString) : ShortString
5650:  Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5651:  Function CharCountS( const S : ShortString; C : AnsiChar) : Byte
5652:  Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5653:  Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5654:  Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5655:  Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5656:  Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5657:  Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5658:  Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
      Margin:Cardinal;PadToMargin:Boolean)
```

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

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

```
5812: Function GeometricMean16( const Data, NData : Integer) : Double
5813: Function HarmonicMeanS( const Data : array of Double) : Double
5814: Function HarmonicMean16( const Data, NData : Integer) : Double
5815: Function Largest( const Data : array of Double; K : Integer) : Double
5816: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5817: Function MedianS( const Data : array of Double) : Double
5818: Function Median16( const Data, NData : Integer) : Double
5819: Function Mode( const Data : array of Double) : Double
5820: Function Mode16( const Data, NData : Integer) : Double
5821: Function Percentile( const Data : array of Double; K : Double) : Double
5822: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5823: Function PercentRank( const Data : array of Double; X : Double) : Double
5824: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5825: Function Permutations( Number, NumberChosen : Integer) : Extended
5826: Function Combinations( Number, NumberChosen : Integer) : Extended
5827: Function FactorialS( N : Integer) : Extended
5828: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5829: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5830: Function Smallest( const Data : array of Double; K : Integer) : Double
5831: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5832: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5833: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5834:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5835:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5836: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5837: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5838: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5839: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5840: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5841: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5842: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5843: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5844: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5845: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5846: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5847: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5848: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5849: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5850: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5851: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5852: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5853: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5854: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5855: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5856: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5857: Function NormSDist( Z : Single) : Single
5858: Function NormSInv( Probability : Single) : Single
5859: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5860: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5861: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5862: Function Erfc( X : Single) : Single
5863: Function GammaLn( X : Single) : Single
5864: Function LargestSort( const Data : array of Double; K : Integer) : Double
5865: Function SmallestSort( const Data : array of double; K : Integer) : Double
5866:
5867: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5868:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5869:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5870:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5871:  Function DefaultMergeName( MergeNum : Integer) : string
5872:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5873:
5874: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5875: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5876:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5877:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5878:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5879:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5880:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5881:  Function LunarPhase( UT : TStDateTimeRec) : Double
5882:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5883:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5884:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5885:  Function FullMoon( D : TStDate) : TStLunarRecord
5886:  Function LastQuarter( D : TStDate) : TStLunarRecord
5887:  Function NewMoon( D : TStDate) : TStLunarRecord
5888:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5889:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5890:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5891:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5892:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5893:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5894:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5895:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5896:  Function SiderealTime( UT : TStDateTimeRec) : Double
5897:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5898:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
```

```
5899:  Function SEaster( Y, Epoch : Integer) : TStDate
5900:  Function DateTimeToAJD( D : TDateTime) : Double
5901:  Function HoursMin( RA : Double) : ShortString
5902:  Function DegsMin( DC : Double) : ShortString
5903:  Function AJDToDateTime( D : Double) : TDateTime
5904:
5905: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5906:  Function CurrentDate : TStDate
5907:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5908:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5909:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5910:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5911:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5912:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5913:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5914:  Function WeekOfYear( Julian : TStDate) : Byte
5915:  Function AstJulianDate( Julian : TStDate) : Double
5916:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5917:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5918:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5919:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5920:  Function StIsLeapYear( Year : Integer) : Boolean
5921:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5922:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5923:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5924:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5925:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5926:  Function CurrentTime : TStTime
5927:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5928:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5929:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5930:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5931:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5932:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5933:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5934:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5935:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5936:  Function StDateToDateTime( D : TStDate) : TDateTime
5937:  Function StTimeToDateTime( T : TStTime) : TDateTime
5938:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5939:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5940:
5941: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5942: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5943:  Function MonthToString( const Month : Integer) : string
5944:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5945:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5946:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5947:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
5948:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5949:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5950:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5951:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5952:  Function TimeStringToStTime( const Picture, S : string) : TStTime
5953:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5954:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5955:  Function DateStringIsBlank( const Picture, S : string) : Boolean
5956:  Function InternationalDate( ForceCentury : Boolean) : string
5957:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
5958:  Function InternationalTime( ShowSeconds : Boolean) : string
5959:  Procedure ResetInternationalInfo
5960:
5961: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5962:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
5963:  Function AnsiUpperCaseShort32( const S : string) : string
5964:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
5965:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
5966:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
5967:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5968:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
5969:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
5970:  Function Upcase( C : AnsiChar) : AnsiChar
5971:  Function LoCase( C : AnsiChar) : AnsiChar
5972:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
5973:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
5974:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
5975:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5976:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5977:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
5978:  Procedure RaiseContainerError( Code : longint)
5979:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
5980:  Function ProductOverflow( A, B : LongInt) : Boolean
5981:  Function StNewStr( S : string) : PShortString
5982:  Procedure StDisposeStr( PS : PShortString)
5983:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
5984:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
5985:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
5986:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
5987:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
```

```
5988:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
5989:
5990: procedure SIRegister_usvd(CL: TPSPascalCompiler);
5991: begin
5992:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
5993:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
5994:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
5995:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
5996:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
5997: end;
5998:
5999: //**********unit unit ; StMath Package of SysTools*****************************************
6000: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
6001: Function PowerS( Base, Exponent : Extended) : Extended
6002: Function StInvCos( X : Double) : Double
6003: Function StInvSin( Y : Double) : Double
6004: Function StInvTan2( X, Y : Double) : Double
6005: Function StTan( A : Double) : Double
6006: Procedure DumpException;  //unit StExpEng;
6007: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6008:
6009: //**********unit unit ; StCRC Package of SysTools*****************************************
6010: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6011: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6012: Function Adler32OfFile( FileName : AnsiString) : LongInt
6013: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6014: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6015: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6016: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6017: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6018: Function Crc32OfFile( FileName : AnsiString) : LongInt
6019: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6020: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6021: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6022: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6023: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6024: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6025:
6026: //**********unit unit ; StBCD Package of SysTools*****************************************
6027: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6028: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6029: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6030: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6031: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6032: Function NegBcd( const B : TbcdS) : TbcdS
6033: Function AbsBcd( const B : TbcdS) : TbcdS
6034: Function FracBcd( const B : TbcdS) : TbcdS
6035: Function IntBcd( const B : TbcdS) : TbcdS
6036: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6037: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6038: Function ValBcd( const S : string) : TbcdS
6039: Function LongBcd( L : LongInt) : TbcdS
6040: Function ExtBcd( E : Extended) : TbcdS
6041: Function ExpBcd( const B : TbcdS) : TbcdS
6042: Function LnBcd( const B : TbcdS) : TbcdS
6043: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6044: Function PowBcd( const B, E : TbcdS) : TbcdS
6045: Function SqrtBcd( const B : TbcdS) : TbcdS
6046: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6047: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6048: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6049: Function IsIntBcd( const B : TbcdS) : Boolean
6050: Function TruncBcd( const B : TbcdS) : LongInt
6051: Function BcdExt( const B : TbcdS) : Extended
6052: Function RoundBcd( const B : TbcdS) : LongInt
6053: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6054: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6055: Function FormatBcd( const Format : string; const B : TbcdS) : string
6056: Function StrGeneralBcd( const B : TbcdS) : string
6057: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6058: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6059:
6060: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6061: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6062: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6063: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6064: Function StDeEscape( const EscStr : AnsiString) : Char
6065: Function StDoEscape( Delim : Char) : AnsiString
6066: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6067: Function AnsiHashText( const S : string; Size : Integer) : Integer
6068: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6069: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6070: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6071:
6072: //**********unit unit ; StNetCon Package of SysTools*****************************************
6073:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6074:     Constructor Create( AOwner : TComponent)
6075:     Function Connect : DWord
6076:     Function Disconnect : DWord
```

```
6077:     RegisterProperty('Password', 'String', iptrw);
6078:     Property('UserName', 'String', iptrw);
6079:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6080:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6081:     Property('LocalDevice', 'String', iptrw);
6082:     Property('ServerName', 'String', iptrw);
6083:     Property('ShareName', 'String', iptrw);
6084:     Property('OnConnect', 'TNotifyEvent', iptrw);
6085:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6086:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6087:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6088:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6089:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6090:   end;
6091: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6092: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6093: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6094: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6095: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6096: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6097: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6098: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6099: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6100: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6101: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6102: Function SuspendThread( hThread : THandle) : DWORD
6103: Function ResumeThread( hThread : THandle) : DWORD
6104: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6105: Function GetCurrentThread : THandle
6106: Procedure ExitThread( dwExitCode : DWORD)
6107: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6108: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6109: Procedure EndThread(ExitCode: Integer);
6110: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6111: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6112: Procedure FreeProcInstance( Proc : FARPROC)
6113: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6114: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6115: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6116: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6117: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6118: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6119: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6120: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6121: Function CurrentParallelJobInfo : TParallelJobInfo
6122: Function ObtainParallelJobInfo : TParallelJobInfo
6123: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6124: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL');
6125: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL');
6126: Function DeviceIoControl( hDevice : THandle; dwIoControlCode : DWORD; lpInBuffer : TObject; nInBufferSize
      : DWORD; lpOutBuffer: TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped :
      TOverlapped) : BOOL');
6127: Function SetFileTime( hFile : THandle; lpCreationTime, lpLastAccessTime, lpLastWriteTime : TFileTime) :
      BOOL');
6128: Function DuplicateHandle( hSourceProcessHandle, hSourceHandle, hTargetProcessHandle : THandle;
      lpTargetHandle : THandle; dwDesiredAccess : DWORD; bInheritHandle : BOOL; dwOptions : DWORD) : BOOL');
6129: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL');
6130: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL');
6131:
6132: ************************************************unit uPSI_JclMime;
6133:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6134:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6135:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6136:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6137:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6138:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6139:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6140:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6141:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6142:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6143:
6144: ************************************************unit uPSI_JclPrint;
6145:  Procedure DirectPrint( const Printer, Data : string)
6146:  Procedure SetPrinterPixelsPerInch
6147:  Function GetPrinterResolution : TPoint
6148:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6149:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6150:
6151:
6152: //***************************************unit uPSI_ShLwApi;***********************************
6153:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6154:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6155:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6156:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6157:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6158:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
```

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

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

```
6336:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6337:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6338:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6339:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6340:  SHREGDEL_DEFAULT','LongWord( $00000000);
6341:  SHREGDEL_HKCU','LongWord( $00000001);
6342:  SHREGDEL_HKLM','LongWord( $00000010);
6343:  SHREGDEL_BOTH','LongWord( $00000011);
6344:  SHREGENUM_DEFAULT','LongWord( $00000000);
6345:  SHREGENUM_HKCU','LongWord( $00000001);
6346:  SHREGENUM_HKLM','LongWord( $00000010);
6347:  SHREGENUM_BOTH','LongWord( $00000011);
6348:  SHREGSET_HKCU','LongWord( $00000001);
6349:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6350:  SHREGSET_HKLM','LongWord( $00000004);
6351:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6352:  TSHRegDelFlags', 'DWORD
6353:  TSHRegEnumFlags', 'DWORD
6354:  HUSKEY', 'THandle
6355:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6356:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6357:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6358:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6359:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6360:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6361:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6362:  ASSOCF_VERIFY','LongWord( $00000040);
6363:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6364:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6365:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6366:  ASSOCF', 'DWORD
6367:  ASSOCSTR_COMMAND','LongInt'( 1);
6368:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
6369:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
6370:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6371:  ASSOCSTR_NOOPEN','LongInt'( 5);
6372:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6373:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
6374:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6375:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
6376:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6377:  ASSOCSTR_INFOTIP','LongInt'( 11);
6378:  ASSOCSTR_MAX','LongInt'( 12);
6379:  ASSOCSTR', 'DWORD
6380:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6381:  ASSOCKEY_APP','LongInt'( 2);
6382:  ASSOCKEY_CLASS','LongInt'( 3);
6383:  ASSOCKEY_BASECLASS','LongInt'( 4);
6384:  ASSOCKEY_MAX','LongInt'( 5);
6385:  ASSOCKEY', 'DWORD
6386:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6387:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6388:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6389:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6390:  ASSOCDATA_MAX','LongInt'( 5);
6391:  ASSOCDATA', 'DWORD
6392:  ASSOCENUM_NONE','LongInt'( 0);
6393:  ASSOCENUM', 'DWORD
6394:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6395:  SHACF_DEFAULT $00000000);
6396:  SHACF_FILESYSTEM','LongWord( $00000001);
6397:  SHACF_URLHISTORY','LongWord( $00000002);
6398:  SHACF_URLMRU','LongWord( $00000004);
6399:  SHACF_USETAB','LongWord( $00000008);
6400:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6401:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6402:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6403:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6404:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6405:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6406:  Procedure SHSetThreadRef( punk : IUnknown)
6407:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6408:  CTF_INSIST','LongWord( $00000001);
6409:  CTF_THREAD_REF','LongWord( $00000002);
6410:  CTF_PROCESS_REF','LongWord( $00000004);
6411:  CTF_COINIT','LongWord( $00000008);
6412:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6413:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6414:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6415:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6416:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6417:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6418:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6419:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6420:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6421:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6422:  Function SetRectEmpty( var lprc : TRect) : BOOL
6423:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6424:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
```

```
6425:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6426:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6427:
6428:  Function InitializeFlatSB( hWnd : HWND) : Bool
6429:  Procedure UninitializeFlatSB( hWnd : HWND)
6430:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6431:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6432:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6433:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6434:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6435:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6436:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6437:
6438:
6439: //  ***************************************** 204 unit uPSI_ShellAPI;
6440: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6441: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6442: Procedure DragFinish( Drop : HDROP)
6443: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6444: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6445: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6446: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6447: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6448: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6449: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6450: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6451: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6452: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6453: Procedure SHFreeNameMappings( hNameMappings : THandle)
6454:
6455:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6456:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6457:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6458:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6459:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6460:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6461:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6462:  Function SimpleXMLEncode( const S : string) : string
6463:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6464:  Function XMLEncode( const S : string) : string
6465:  Function XMLDecode( const S : string) : string
6466:  Function EntityEncode( const S : string) : string
6467:  Function EntityDecode( const S : string) : string
6468:
6469: procedure RIRegister_CPort_Routines(S: TPSExec);
6470:  Procedure EnumComPorts( Ports : TStrings)
6471:  Procedure ListComPorts( Ports : TStrings)
6472:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6473:  Function GetComPorts: TStringlist;
6474:  Function StrToBaudRate( Str : string) : TBaudRate
6475:  Function StrToStopBits( Str : string) : TStopBits
6476:  Function StrToDataBits( Str : string) : TDataBits
6477:  Function StrToParity( Str : string) : TParityBits
6478:  Function StrToFlowControl( Str : string) : TFlowControl
6479:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6480:  Function StopBitsToStr( StopBits : TStopBits) : string
6481:  Function DataBitsToStr( DataBits : TDataBits) : string
6482:  Function ParityToStr( Parity : TParityBits) : string
6483:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6484:  Function ComErrorsToStr( Errors : TComErrors) : String
6485:
6486:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6487:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6488:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6489:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6490:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6491:  Function GetMessagePos : DWORD
6492:  Function GetMessageTime : Longint
6493:  Function GetMessageExtraInfo : Longint
6494:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6495:  Procedure JAddToRecentDocs( const Filename : string)
6496:  Procedure ClearRecentDocs
6497:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6498:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6499:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6500:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6501:  Function RecycleFile( FileToRecycle : string) : Boolean
6502:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6503:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6504:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6505:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6506:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6507:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6508:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
```

```
6509:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
       dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
       lpResumeHandle:DWORD; pszGroupNam
6510:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
       dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
       lpResumeHandle:DWORD; pszGroupName
6511:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6512:
6513: ****************************************** unit uPSI_JclPeImage;
6514:
6515:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6516: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6517:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6518:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
       DWORD) : TJclRebaseImageInfo
6519:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6520:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6521:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6522:  Function PeDoesExportFunction(const FileName:TFileName;const
       FuncName:string;Options:TJclSmartCompOptions):Bool;
6523:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
       ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6524:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
       FunctionName:string;Options:TJclSmartCompOptions):Bool
6525:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
       : string; Options : TJclSmartCompOptions) : Boolean
6526:  Function PeDoesImportLibrary(const FileName:TFileName;const
       LibraryName:string;Recursive:Boolean):Boolean);
6527:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
       Boolean; FullPathName : Boolean) : Boolean
6528:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
       LibraryName:string; IncludeLibNames : Boolean): Boolean
6529:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6530:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6531:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6532:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
       NamesList:TStrings):Bool
6533:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6534:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
       Descript:Bool):Bool;
6535:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6536:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6537:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6538:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6539:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6540:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6541:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
       PImageSectionHeader
6542:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6543:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6544:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
       ___Pointer;
6545:    SIRegister_TJclPeSectionStream(CL);
6546:    SIRegister_TJclPeMapImgHookItem(CL);
6547:    SIRegister_TJclPeMapImgHooks(CL);
6548:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
       NtHeaders:TImageNtHeaders):Boolean
6549:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6550:    Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6551:    TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6552:    TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6553:    TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6554:    TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6555:    TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6556:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
       TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6557:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
       Descript:TJclBorUmDescription):TJclBorUmResult;
6558:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6559:  Function PeBorUnmangleName3( const Name : string) : string;
6560:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6561:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6562:
6563:
6564: //****************** SysTools uPSI_StSystem; ************************************
6565:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6566:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6567:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6568:  //Procedure EnumerateDirectories(const
       StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6569:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
       IncludeItem:TIncludeItemFunc);
6570:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6571:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6572:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6573:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6574:  Function FlushOsBuffers( Handle : Integer) : Boolean
6575:  Function GetCurrentUser : AnsiString
6576:  Function GetDiskClass( Drive : Char) : DiskClass
```

```
6577:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
       SectorsPerCluster:Cardinal):Bool;
6578:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
       DiskSize:Double):Bool;
6579:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
       DiskSize:Comp):Boolean;
6580:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6581:  Function getDiskSpace2(const path: String; index: integer): int64;
6582:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6583:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6584:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6585:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6586:  Function GetLongPath( const APath : AnsiString) : AnsiString
6587:  Function GetMachineName : AnsiString
6588:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6589:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6590:  Function GetShortPath( const APath : AnsiString) : AnsiString
6591:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6592:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6593:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6594:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6595:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6596:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6597:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6598:  Function IsDriveReady( Drive : Char) : Boolean
6599:  Function IsFile( const FileName : AnsiString) : Boolean
6600:  Function IsFileArchive( const S : AnsiString) : Integer
6601:  Function IsFileHidden( const S : AnsiString) : Integer
6602:  Function IsFileReadOnly( const S : AnsiString) : Integer
6603:  Function IsFileSystem( const S : AnsiString) : Integer
6604:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6605:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6606:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6607:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6608:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6609:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6610:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6611:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6612:  Function ValidDrive( Drive : Char) : Boolean
6613:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6614:
6615: //****************************unit uPSI_JclLANMan;*****************************************
6616: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6617:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6618:  Function DeleteAccount( const Servername, Username : string) : Boolean
6619:  Function DeleteLocalAccount( Username : string) : Boolean
6620:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6621:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6622:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6623:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6624:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6625:  Function LocalGroupExists( const Group : string) : Boolean
6626:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6627:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6628:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6629:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6630:  Function IsLocalAccount( const AccountName : string) : Boolean
6631:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6632:  Function GetRandomString( NumChar : cardinal) : string
6633:
6634: //****************************unit uPSI_cUtils;*****************************************
6635: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6636: Function cIsWinNT : boolean
6637:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
       Multitasking:Boolean;
6638:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6639:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
       CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6640:  Function cGetShortName( FileName : string) : string
6641:  Procedure cShowError( Msg : String)
6642:  Function cCommaStrToStr( s : string; formatstr : string) : string
6643:  Function cIncludeQuoteIfSpaces( s : string) : string
6644:  Function cIncludeQuoteIfNeeded( s : string) : string
6645:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6646:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6647:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6648:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6649:  Function cCodeInstoStr( s : string) : string
6650:  Function cStrtoCodeIns( s : string) : string
6651:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6652:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6653:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6654:  Function cPointtoStr( const pt : TPoint) : string
6655:  Function cListtoStr( const List : TStrings) : string
6656:  Function ListtoStr( const List : TStrings) : string
6657:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6658:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
```

```
6659:  Function cGetFileTyp( const FileName : string) : TUnitType
6660:  Function cGetExTyp( const FileName : string) : TExUnitType
6661:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6662:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6663:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6664:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6665:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6666:  Function cGenMakePath( FileName : String) : String;
6667:  Function cGenMakePath2( FileName : String) : String
6668:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6669:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6670:  Function cCalcMod( Count : Integer) : Integer
6671:  Function cGetVersionString( FileName : string) : string
6672:  Function cCheckChangeDir( var Dir : string) : boolean
6673:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6674:  Function cIsNumeric( s : string) : boolean
6675:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6676:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6677:  Function GetFileTyp( const FileName : string) : TUnitType
6678:  Function Atoi(const aStr: string): integer
6679:  Function Itoa(const aint: integer): string
6680:  Function Atof(const aStr: string): double');
6681:  Function Atol(const aStr: string): longint');
6682:
6683:
6684: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6685: begin
6686:   FindClass('TOBJECT'),'EHTTP
6687:   FindClass('TOBJECT'),'EHTTPParser
6688:   //AnsiCharSet', 'set of AnsiChar
6689:   AnsiStringArray', 'array of AnsiString
6690:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6691:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6692:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6693:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6694:    +'CustomMinVersion : Integer; end
6695:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6696:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6697:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6698:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6699:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6700:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6701:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6702:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6703:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6704:    +'nection, hntOrigin, hntKeepAlive )
6705:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6706:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6707:    +' AnsiString; end
6708:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6709:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6710:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6711:   //PHTTPContentLength', '^THTTPContentLength // will not work
6712:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6713:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6714:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6715:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6716:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6717:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6718:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6719:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6720:    +' CustomStr : AnsiString; end
6721:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6722:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6723:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6724:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6725:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6726:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6727:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6728:    +'m; Custom : AnsiString; end
6729:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6730:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6731:    +' Custom : AnsiString; end
6732:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6733:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6734:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6735:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6736:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6737:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6738:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6739:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6740:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6741:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6742:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6743:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6744:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6745:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6746:    +'FieldEnum; List : array of THTTPContentEncoding; end
6747:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
```

```
6748:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6749:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6750:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6751:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6752:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6753:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6754:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6755:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6756:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6757:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6758:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6759:    +'CustomFieldArray; Custom : AnsiString; end
6760:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6761:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6762:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6763:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6764:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6765:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6766:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6767:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6768:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6769:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6770:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6771:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6772:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6773:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6774:   THTTPFixedHeaders','array[0..42] of AnsiString
6775:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6776:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6777:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6778:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6779:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6780:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6781:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6782:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6783:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6784:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6785:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6786:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6787:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6788:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6789:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6790:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6791:    +' THTTPDateField; Age : THTTPAgeField; end
6792:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6793:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6794:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6795:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6796:  Procedure InitHTTPRequest( var A : THTTPRequest)
6797:  Procedure InitHTTPResponse( var A : THTTPResponse)
6798:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6799:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6800:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6801:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6802:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6803:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6804:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6805:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6806:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6807:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6808:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6809:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6810:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6811:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6812:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6813:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6814:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6815:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6816:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6817:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6818:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6819:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6820:   THTTPStringOption', '( hsoNone )
6821:   THTTPStringOptions', 'set of THTTPStringOption
6822:   FindClass('TOBJECT'),'TAnsiStringBuilder
6823:
6824:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6825:  Procedure BuildStrHTTPContentLengthValue(const
      A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6826:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
      B:TAnsiStringBuilder;P:THTTPStringOptions)
6827:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
      P:THTTPStringOptions)
6828:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6829:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6830:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
```

```
6831:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
       P:THTTPStringOptions);
6832:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
       TAnsiStringBuilder; const P : THTTPStringOptions)
6833:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6834:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6835:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6836:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6837:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
       P:THTTPStringOptions);
6838:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6839:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
       B:TAnsiStringBuilder;const P:THTTPStringOptions)
6840:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6841:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
       : THTTPStringOptions)
6842:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions)
6843:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
       : THTTPStringOptions)
6844:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6845:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
       : THTTPStringOptions)
6846:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions);
6847:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
       THTTPStringOptions)
6848:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6849:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
       P:THTTPStringOptions);
6850:  Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
       THTTPStringOptions)
6851:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
       TAnsiStringBuilder; const P : THTTPStringOptions)
6852:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
       THTTPStrOptions);
6853:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions);
6854:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
       P:THTTPStringOptions);
6855:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6856:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6857:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6858:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6859:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6860:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6861:  Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
       Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6862:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6863:   SIRegister_THTTPParser(CL);
6864:   FindClass('TOBJECT'),'THTTPContentDecoder
6865:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6866:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6867:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6868:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
6869:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6870:   SIRegister_THTTPContentDecoder(CL);
6871:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6872:   FindClass('TOBJECT'),'THTTPContentReader
6873:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6874:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
       LogLevel:Int;
6875:   SIRegister_THTTPContentReader(CL);
6876:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
6877:   FindClass('TOBJECT'),'THTTPContentWriter
6878:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6879:   SIRegister_THTTPContentWriter(CL);
6880:  Procedure SelfTestcHTTPUtils
6881: end;
6882:
6883: (*----------------------------------------------------------------------------*)
6884: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6885: begin
6886:  'TLSLibraryVersion','String '1.00
6887:  'TLSError_None','LongInt'( 0);
6888:  'TLSError_InvalidBuffer','LongInt'( 1);
6889:  'TLSError_InvalidParameter','LongInt'( 2);
6890:  'TLSError_InvalidCertificate','LongInt'( 3);
6891:  'TLSError_InvalidState','LongInt'( 4);
6892:  'TLSError_DecodeError','LongInt'( 5);
6893:  'TLSError_BadProtocol','LongInt'( 6);
```

```
6894:  Function TLSErrorMessage( const TLSError : Integer) : String
6895:   SIRegister_ETLSError(CL);
6896:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6897:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6898:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6899:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6900:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6901:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6902:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6903:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6904:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6905:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6906:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6907:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6908:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6909:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6910:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6911:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6912:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6913:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6914:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6915:   PTLSRandom', '^TTLSRandom // will not work
6916:  Procedure InitTLSRandom( var Random : TTLSRandom)
6917:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6918:  'TLSSessionIDMaxLen','LongInt'( 32);
6919:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6920:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6921:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6922:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6923:    +'; Signature : TTLSSignatureAlgorithm; end
6924:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6925:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6926:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6927:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6928:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6929:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6930:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6931:    +'nteger; Supported : Boolean; end
6932:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6933:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
6934:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6935:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6936:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6937:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6938:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6939:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6940:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6941:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6942:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
6943:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
      Size:Integer):AnsiString
6944:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6945:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6946:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
      ClientRandom : AnsiString; const Size : Integer) : AnsiString
6947:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6948:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6949:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6950:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:AnsiString) : AnsiString
6951:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6952:    +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6953:    +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6954:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
      IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6955:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
      ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
6956:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
6957:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
6958:  Procedure SelfTestcTLSUtils
6959: end;
6960:
6961: (*----------------------------------------------------------------------------*)
6962: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6963: begin
6964:   sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
      integer; disks : integer; mx : integer; my : integer; end
6965:  // pBoard', '^tBoard // will not work
6966:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6967:  Function rCheckMove( color : byte; cx, cy : integer) : integer
6968:  //Function rDoStep( data : pBoard) : word
6969:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6970: end;
6971:
6972: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
6973: begin
```

```
6974: Function InEditMode( ADataset : TDataset) : Boolean
6975: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
6976: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6977: Function GetFieldText( AField : TField) : String
6978: end;
6979:
6980: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
6981: begin
6982:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6983:   TMyPrintRange', '( prAll, prSelected )
6984:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
6985:    +'ded, ssDateTime, ssTime, ssCustom )
6986:   TSortDirection', '( sdAscending, sdDescending )
6987:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
6988:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
6989:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
6990:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
6991:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
6992:   SIRegister_TSortOptions(CL);
6993:   SIRegister_TPrintOptions(CL);
6994:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
6995:   SIRegister_TSortedList(CL);
6996:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
6997:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
6998:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
6999:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7000:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7001:   SIRegister_TFontSetting(CL);
7002:   SIRegister_TFontList(CL);
7003:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
7004:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7005:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7006:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7007:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
7008:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7009:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7010:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7011:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
7012:    +'r; var SortStyle : TSortStyle)
7013:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
7014:    +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7015:   SIRegister_TSortGrid(CL);
7016:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
7017:  Function NormalCompare( const Str1, Str2 : String) : Integer
7018:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
7019:  Function NumericCompare( const Str1, Str2 : String) : Integer
7020:  Function TimeCompare( const Str1, Str2 : String) : Integer
7021:  //Function Compare( Item1, Item2 : Pointer) : Integer
7022: end;
7023:
7024: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
7025:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
7026:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
7027:  Procedure IBDataBaseError
7028:  Function StatusVector : PISC_STATUS
7029:  Function StatusVectorArray : PStatusVector
7030:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
7031:  Function StatusVectorAsText : string
7032:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7033:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7034:
7035:
7036: //****************************unit uPSI_BoldUtils;****************************************
7037:  Function CharCount( c : char; const s : string) : integer
7038:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
7039:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7040:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7041:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7042:  Function BoldTrim( const S : string) : string
7043:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
7044:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7045:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7046:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7047:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7048:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7049:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7050:  Function CapitalisedToSpaced( Capitalised : String) : String
7051:  Function SpacedToCapitalised( Spaced : String) : String
7052:  Function BooleanToString( BoolValue : Boolean) : String
7053:  Function StringToBoolean( StrValue : String) : Boolean
7054:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7055:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7056:  Function StringListToVarArray( List : TStringList) : variant
7057:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7058:  Function GetComputerNameStr : string
7059:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7060:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7061:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7062:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
```

```
7063:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7064:  Procedure EnsureTrailing( var Str : String; ch : char)
7065:  Function BoldDirectoryExists( const Name : string) : Boolean
7066:  Function BoldForceDirectories( Dir : string) : Boolean
7067:  Function BoldRootRegistryKey : string
7068:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7069:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7070:  Function LogicalAnd( A, B : Integer) : Boolean
7071:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7072:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7073:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7074:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7075:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7076:  Function IsFirstInstance : Boolean
7077:  Procedure ActivateFirst( AString : PChar)
7078:  Procedure ActivateFirstCommandLine
7079:  function MakeAckPkt(const BlockNumber: Word): string;
7080:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7081:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7082:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7083:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7084:  function IdStrToWord(const Value: String): Word;
7085:  function IdWordToStr(const Value: Word): WordStr;
7086:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7087:  Function CPUFeatures : TCPUFeatures
7088:
7089: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7090: begin
7091:   AddTypeS('TXRTLBitIndex', 'Integer
7092:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7093:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7094:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7095:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7096:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7097:  Function XRTLSwapHiLo16( X : Word) : Word
7098:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7099:  Function XRTLSwapHiLo64( X : Int64) : Int64
7100:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7101:  Function XRTLROR32( A, S : Cardinal) : Cardinal
7102:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7103:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7104:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7105:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7106:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7107:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7108:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7109:  Function XRTLPopulation( A : Cardinal) : Cardinal
7110: end;
7111:
7112: Function XRTLURLDecode( const ASrc : WideString) : WideString
7113: Function XRTLURLEncode( const ASrc : WideString) : WideString
7114: Function XRTLURINormalize( const AURI : WideString) : WideString
7115: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7116: Function XRTLExtractLongPathName(APath: string): string;
7117:
7118: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7119: begin
7120:   AddTypeS('Int8', 'ShortInt
7121:   AddTypeS('Int16', 'SmallInt
7122:   AddTypeS('Int32', 'LongInt
7123:   AddTypeS('UInt8', 'Byte
7124:   AddTypeS('UInt16', 'Word
7125:   AddTypeS('UInt32', 'LongWord
7126:   AddTypeS('UInt64', 'Int64
7127:   AddTypeS('Word8', 'UInt8
7128:   AddTypeS('Word16', 'UInt16
7129:   AddTypeS('Word32', 'UInt32
7130:   AddTypeS('Word64', 'UInt64
7131:   AddTypeS('LargeInt', 'Int64
7132:   AddTypeS('NativeInt', 'Integer
7133:   AddTypeS('NativeUInt', 'Cardinal
7134:   Const('BitsPerByte','LongInt'( 8);
7135:   Const('BitsPerWord','LongInt'( 16);
7136:   Const('BitsPerLongWord','LongInt'( 32);
7137:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7138:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7139:  Function MinI( const A, B : Integer) : Integer
7140:  Function MaxI( const A, B : Integer) : Integer
7141:  Function MinC( const A, B : Cardinal) : Cardinal
7142:  Function MaxC( const A, B : Cardinal) : Cardinal
7143:  Function SumClipI( const A, I : Integer) : Integer
7144:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7145:  Function InByteRange( const A : Int64) : Boolean
7146:  Function InWordRange( const A : Int64) : Boolean
7147:  Function InLongWordRange( const A : Int64) : Boolean
7148:  Function InShortIntRange( const A : Int64) : Boolean
7149:  Function InSmallIntRange( const A : Int64) : Boolean
```

```
7150:  Function InLongIntRange( const A : Int64) : Boolean
7151:   AddTypeS('Bool8', 'ByteBool
7152:   AddTypeS('Bool16', 'WordBool
7153:   AddTypeS('Bool32', 'LongBool
7154:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7155:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7156:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7157:  Const('MinSingle','Single').setExtended( 1.5E-45);
7158:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7159:  Const('MinDouble','Double').setExtended( 5.0E-324);
7160:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7161:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7162:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7163:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7164:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7165:  Function MinF( const A, B : Float) : Float
7166:  Function MaxF( const A, B : Float) : Float
7167:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7168:  Function InSingleRange( const A : Float) : Boolean
7169:  Function InDoubleRange( const A : Float) : Boolean
7170:  Function InCurrencyRange( const A : Float) : Boolean;
7171:  Function InCurrencyRange1( const A : Int64) : Boolean;
7172:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7173:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7174:  Function FloatIsInfinity( const A : Extended) : Boolean
7175:  Function FloatIsNaN( const A : Extended) : Boolean
7176:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7177:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7178:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7179:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7180:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7181:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7182:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7183:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7184:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7185:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7186:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7187:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7188:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7189:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7190:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7191:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7192:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7193:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7194:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7195:  Function SetBitScanForward( const Value : LongWord) : Integer;
7196:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7197:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7198:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7199:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7200:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7201:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7202:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7203:  Function cReverseBits( const Value : LongWord) : LongWord;
7204:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7205:  Function cSwapEndian( const Value : LongWord) : LongWord
7206:  Function cTwosComplement( const Value : LongWord) : LongWord
7207:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7208:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7209:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7210:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7211:  Function cBitCount( const Value : LongWord) : LongWord
7212:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7213:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7214:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7215:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7216:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7217:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7218:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7219:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7220:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7221: //  AddTypeS('CharSet', 'set of AnsiChar
7222:   AddTypeS('CharSet', 'set of Char       //!!!
7223:   AddTypeS('AnsiCharSet', 'TCharSet
7224:   AddTypeS('ByteSet', 'set of Byte
7225:   AddTypeS('AnsiChar', 'Char
7226:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7227:  Function AsByteSet( const C : array of Byte) : ByteSet
7228:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7229:  Procedure ClearCharSet( var C : CharSet)
7230:  Procedure FillCharSet( var C : CharSet)
7231:  Procedure ComplementCharSet( var C : CharSet)
7232:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7233:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7234:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7235:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7236:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7237:  Function IsSubSet( const A, B : CharSet) : Boolean
7238:  Function IsEqual( const A, B : CharSet) : Boolean
```

```
7239:  Function IsEmpty( const C : CharSet) : Boolean
7240:  Function IsComplete( const C : CharSet) : Boolean
7241:  Function cCharCount( const C : CharSet) : Integer
7242:  Procedure ConvertCaseInsensitive( var C : CharSet)
7243:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7244:  Function IntRangeLength( const Low, High : Integer) : Int64
7245:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7246:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7247:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7248:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7249:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7250:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7251:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7252:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7253:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7254:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7255:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7256:   AddTypeS('UnicodeChar', 'WideChar
7257:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7258:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7259:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7260:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7261:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7262:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7263:  Function cSgn( const A : LongInt) : Integer;
7264:  Function cSgn1( const A : Int64) : Integer;
7265:  Function cSgn2( const A : Extended) : Integer;
7266:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7267:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7268:  Function WideCharToInt( const A : WideChar) : Integer
7269:  Function CharToInt( const A : Char) : Integer
7270:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7271:  Function IntToWideChar( const A : Integer) : WideChar
7272:  Function IntToChar( const A : Integer) : Char
7273:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7274:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7275:  Function IsHexChar( const Ch : Char) : Boolean
7276:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7277:  Function HexWideCharToInt( const A : WideChar) : Integer
7278:  Function HexCharToInt( const A : Char) : Integer
7279:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7280:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7281:  Function IntToUpperHexChar( const A : Integer) : Char
7282:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7283:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
7284:  Function IntToLowerHexChar( const A : Integer) : Char
7285:  Function IntToStringA( const A : Int64) : AnsiString
7286:  Function IntToStringW( const A : Int64) : WideString
7287:  Function IntToString( const A : Int64) : String
7288:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7289:  Function UIntToStringW( const A : NativeUInt) : WideString
7290:  Function UIntToString( const A : NativeUInt) : String
7291:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7292:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7293:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7294:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7295:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7296:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7297:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7298:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7299:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7300:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7301:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7302:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7303:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7304:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7305:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7306:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7307:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7308:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7309:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7310:  Function StringToInt64A( const S : AnsiString) : Int64
7311:  Function StringToInt64W( const S : WideString) : Int64
7312:  Function StringToInt64( const S : String) : Int64
7313:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7314:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7315:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7316:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7317:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7318:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7319:  Function StringToIntA( const S : AnsiString) : Integer
7320:  Function StringToIntW( const S : WideString) : Integer
7321:  Function StringToInt( const S : String) : Integer
7322:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7323:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7324:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7325:  Function StringToLongWordA( const S : AnsiString) : LongWord
7326:  Function StringToLongWordW( const S : WideString) : LongWord
7327:  Function StringToLongWord( const S : String) : LongWord
```

```
7328:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7329:  Function HexToUIntW( const S : WideString) : NativeUInt
7330:  Function HexToUInt( const S : String) : NativeUInt
7331:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7332:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7333:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7334:  Function HexToLongWordA( const S : AnsiString) : LongWord
7335:  Function HexToLongWordW( const S : WideString) : LongWord
7336:  Function HexToLongWord( const S : String) : LongWord
7337:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7338:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7339:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7340:  Function OctToLongWordA( const S : AnsiString) : LongWord
7341:  Function OctToLongWordW( const S : WideString) : LongWord
7342:  Function OctToLongWord( const S : String) : LongWord
7343:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7344:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7345:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7346:  Function BinToLongWordA( const S : AnsiString) : LongWord
7347:  Function BinToLongWordW( const S : WideString) : LongWord
7348:  Function BinToLongWord( const S : String) : LongWord
7349:  Function FloatToStringA( const A : Extended) : AnsiString
7350:  Function FloatToStringW( const A : Extended) : WideString
7351:  Function FloatToString( const A : Extended) : String
7352:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7353:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7354:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7355:  Function StringToFloatA( const A : AnsiString) : Extended
7356:  Function StringToFloatW( const A : WideString) : Extended
7357:  Function StringToFloat( const A : String) : Extended
7358:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7359:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7360:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7361:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
      PadChar: AnsiChar) : AnsiString
7362:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7363:  unit uPSI_cFundamentUtils;
7364:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7365:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7366:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7367:  Const('CCHARSET','Stringb64_XXEncode);
7368:  Const('CHEXSET','String'0123456789ABCDEF
7369:  Const('HEXDIGITS','String'0123456789ABCDEF
7370:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7371:  Const('DIGISET','String'0123456789
7372:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7373:  Const('DIGISET2','TCharset').SetSet('0123456789'
7374:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7375:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7376:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7377:  Const('NUMBERS','String'0123456789');
7378:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7379:  Function CharSetToStr( const C : CharSet) : AnsiString
7380:  Function StrToCharSet( const S : AnsiString) : CharSet
7381:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7382:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7383:  Function UUDecode( const S : AnsiString) : AnsiString
7384:  Function XXDecode( const S : AnsiString) : AnsiString
7385:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7386:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7387:  Function InterfaceToStrW( const I : IInterface) : WideString
7388:  Function InterfaceToStr( const I : IInterface) : String
7389:  Function ObjectClassName( const O : TObject) : String
7390:  Function ClassClassName( const C : TClass) : String
7391:  Function ObjectToStr( const O : TObject) : String
7392:  Function ObjectToString( const O : TObject) : String
7393:  Function CharSetToStr( const C : CharSet) : AnsiString
7394:  Function StrToCharSet( const S : AnsiString) : CharSet
7395:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7396:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7397:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7398:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7399:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7400:  Const('Bytes1KB','LongInt'( 1024);
7401:   SIRegister_IInterface(CL);
7402:  Procedure SelfTestCFundamentUtils
7403:
7404: Function CreateSchedule : IJclSchedule
7405: Function NullStamp : TTimeStamp
7406: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7407: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7408: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7409:
7410: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7411: begin
7412:  AddTypeS('TFunc', 'function(X : Float) : Float;
```

```
7413: Function InitGraphics( Width, Height : Integer) : Boolean
7414: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7415: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7416: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7417: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7418: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7419: Procedure SetGraphTitle( Title : String)
7420: Procedure SetOxTitle( Title : String)
7421: Procedure SetOyTitle( Title : String)
7422: Function GetGraphTitle : String
7423: Function GetOxTitle : String
7424: Function GetOyTitle : String
7425: Procedure PlotOxAxis( Canvas : TCanvas)
7426: Procedure PlotOyAxis( Canvas : TCanvas)
7427: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7428: Procedure WriteGraphTitle( Canvas : TCanvas)
7429: Function SetMaxCurv( NCurv : Byte) : Boolean
7430: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7431: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7432: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7433: Procedure SetCurvStep( CurvIndex, Step : Integer)
7434: Function GetMaxCurv : Byte
7435: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7436: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7437: Function GetCurvLegend( CurvIndex : Integer) : String
7438: Function GetCurvStep( CurvIndex : Integer) : Integer
7439: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7440: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7441: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7442: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7443: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7444: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7445: Function Xpixel( X : Float) : Integer
7446: Function Ypixel( Y : Float) : Integer
7447: Function Xuser( X : Integer) : Float
7448: Function Yuser( Y : Integer) : Float
7449: end;
7450:
7451: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7452: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7453: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7454: Procedure FFT_Integer_Cleanup
7455: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7456: //unit uPSI_JclStreams;
7457: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7458: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7459: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7460: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7461:
7462: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7463: begin
7464:  FindClass('TOBJECT'),'EInvalidDest
7465:  FindClass('TOBJECT'),'EFCantMove
7466:  Procedure fmxCopyFile( const FileName, DestName : string)
7467:  Procedure fmxMoveFile( const FileName, DestName : string)
7468:  Function fmxGetFileSize( const FileName : string) : LongInt
7469:  Function fmxFileDateTime( const FileName : string) : TDateTime
7470:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7471:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7472: end;
7473:
7474: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7475: begin
7476:  SIRegister_IFindFileIterator(CL);
7477:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7478: end;
7479:
7480: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7481: begin
7482:  Function SkipWhite( cp : PChar) : PChar
7483:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7484:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7485:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7486: end;
7487:
7488: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7489: begin
7490:   SIRegister_TStringHashMapTraits(CL);
7491:  Function CaseSensitiveTraits : TStringHashMapTraits
7492:  Function CaseInsensitiveTraits : TStringHashMapTraits
7493:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7494:    +'e; Right : PHashNode; end
7495:   //PHashArray', '^THashArray // will not work
7496:   SIRegister_TStringHashMap(CL);
7497:  THashValue', 'Cardinal
7498:  Function StrHash( const s : string) : THashValue
7499:  Function TextHash( const s : string) : THashValue
7500:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7501:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
```

```
7502:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7503:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7504:   SIRegister_TCaseSensitiveTraits(CL);
7505:   SIRegister_TCaseInsensitiveTraits(CL);
7506:
7507:
7508: //***************************************************************************unit uPSI_umath;
7509:  Function uExpo( X : Float) : Float
7510:  Function uExp2( X : Float) : Float
7511:  Function uExp10( X : Float) : Float
7512:  Function uLog( X : Float) : Float
7513:  Function uLog2( X : Float) : Float
7514:  Function uLog10( X : Float) : Float
7515:  Function uLogA( X, A : Float) : Float
7516:  Function uIntPower( X : Float; N : Integer): Float
7517:  Function uPower( X, Y : Float) : Float
7518:  Function SgnGamma( X : Float) : Integer
7519:  Function Stirling( X : Float) : Float
7520:  Function StirLog( X : Float) : Float
7521:  Function Gamma( X : Float) : Float
7522:  Function LnGamma( X : Float) : Float
7523:  Function DiGamma( X : Float) : Float
7524:  Function TriGamma( X : Float) : Float
7525:  Function IGamma( X : Float) : Float
7526:  Function JGamma( X : Float) : Float
7527:  Function InvGamma( X : Float) : Float
7528:  Function Erf( X : Float) : Float
7529:  Function Erfc( X : Float) : Float
7530: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7531: { Correlation coefficient between samples X and Y }
7532: function DBeta(A, B, X : Float) : Float;
7533: { Density of Beta distribution with parameters A and B }
7534: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7535: Function Beta(X, Y : Float) : Float
7536: Function Binomial( N, K : Integer) : Float
7537: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7538: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7539: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7540: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7541: Function DNorm( X : Float) : Float
7542:
7543: function DGamma(A, B, X : Float) : Float;
7544: { Density of Gamma distribution with parameters A and B }
7545: function DKhi2(Nu : Integer; X : Float) : Float;
7546: { Density of Khi-2 distribution with Nu d.o.f. }
7547: function DStudent(Nu : Integer; X : Float) : Float;
7548: { Density of Student distribution with Nu d.o.f. }
7549: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7550: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7551: function IBeta(A, B, X : Float) : Float;
7552: { Incomplete Beta function}
7553: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7554:
7555: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7556: begin
7557:  Procedure SetOptAlgo( Algo : TOptAlgo)
7558: procedure SetOptAlgo(Algo : TOptAlgo);
7559: { ----------------------------------------------------------------
7560:   Sets the optimization algorithm according to Algo, which must be
7561:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7562:
7563:  Function GetOptAlgo : TOptAlgo
7564:  Procedure SetMaxParam( N : Byte)
7565:  Function GetMaxParam : Byte
7566:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7567:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7568:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7569:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
      Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7570:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
      MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7571:  Procedure SetMCFile( FileName : String)
7572:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7573:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
      LastPar:Integer;V:TMatrix);
7574: end;
7575:
7576: (*----------------------------------------------------------------------------*)
7577: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7578: begin
7579:  Procedure SaveSimplex( FileName : string)
7580:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7581: end;
7582: (*----------------------------------------------------------------------------*)
7583: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7584: begin
7585:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7586:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7587: end;
```

```
7588:
7589: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7590: begin
7591:  Function LTrim( S : String) : String
7592:  Function RTrim( S : String) : String
7593:  Function uTrim( S : String) : String
7594:  Function StrChar( N : Byte; C : Char) : String
7595:  Function RFill( S : String; L : Byte) : String
7596:  Function LFill( S : String; L : Byte) : String
7597:  Function CFill( S : String; L : Byte) : String
7598:  Function Replace( S : String; C1, C2 : Char) : String
7599:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7600:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7601:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7602:  Function FloatStr( X : Float) : String
7603:  Function IntStr( N : LongInt) : String
7604:  Function uCompStr( Z : Complex) : String
7605: end;
7606:
7607: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7608: begin
7609:  Function uSinh( X : Float) : Float
7610:  Function uCosh( X : Float) : Float
7611:  Function uTanh( X : Float) : Float
7612:  Function uArcSinh( X : Float) : Float
7613:  Function uArcCosh( X : Float) : Float
7614:  Function ArcTanh( X : Float) : Float
7615:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7616: end;
7617:
7618: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7619: begin
7620: type RNG_Type =
7621:   (RNG_MWC,      { Multiply-With-Carry }
7622:    RNG_MT,       { Mersenne Twister }
7623:    RNG_UVAG);    { Universal Virtual Array Generator }
7624:  Procedure SetRNG( RNG : RNG_Type)
7625:  Procedure InitGen( Seed : RNG_IntType)
7626:  Procedure SRand( Seed : RNG_IntType)
7627:  Function IRanGen : RNG_IntType
7628:  Function IRanGen31 : RNG_IntType
7629:  Function RanGen1 : Float
7630:  Function RanGen2 : Float
7631:  Function RanGen3 : Float
7632:  Function RanGen53 : Float
7633: end;
7634:
7635: //  Optimization by Simulated Annealing
7636: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7637: begin
7638:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7639:  Procedure SA_CreateLogFile( FileName : String)
7640:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7641: end;
7642:
7643: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7644: begin
7645:  Procedure InitUVAGbyString( KeyPhrase : string)
7646:  Procedure InitUVAG( Seed : RNG_IntType)
7647:  Function IRanUVAG : RNG_IntType
7648: end;
7649:
7650: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7651: begin
7652:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7653:  Procedure GA_CreateLogFile( LogFileName : String)
7654:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7655: end;
7656:
7657:  TVector', 'array of Float
7658: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7659: begin
7660:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7661:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7662: end;
7663:
7664: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7665: begin
7666:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7667:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7668: end;
7669:
7670: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7671: begin
7672:   FT_Result', 'Integer
7673:   //TDWordptr', '^DWord // will not work
7674:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7675:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7676:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
```

```
7677:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7678:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7679:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7680:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7681:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7682:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7683:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7684:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7685:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7686:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7687:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7688:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7689:    yte; end
7690: end;
7691:
7692:
7693: //************************************ PaintFX*************************
7694: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7695: begin
7696:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7697:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7698:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7699:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7700:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7701:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7702:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7703:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7704:     Procedure Turn( Src, Dst : TBitmap)
7705:     Procedure TurnRight( Src, Dst : TBitmap)
7706:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7707:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7708:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7709:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7710:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7711:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7712:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7713:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7714:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7715:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7716:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7717:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7718:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7719:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7720:     Procedure Emboss( var Bmp : TBitmap)
7721:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7722:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7723:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7724:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7725:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7726:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7727:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7728:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7729:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7730:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7731:     Procedure SemiOpaque( Src, Dst : TBitmap)
7732:     Procedure ShadowDownLeft( const Dst : TBitmap)
7733:     Procedure ShadowDownRight( const Dst : TBitmap)
7734:     Procedure ShadowUpLeft( const Dst : TBitmap)
7735:     Procedure ShadowUpRight( const Dst : TBitmap)
7736:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7737:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7738:     Procedure FlipRight( const Dst : TBitmap)
7739:     Procedure FlipDown( const Dst : TBitmap)
7740:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7741:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7742:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7743:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7744:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7745:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7746:     Procedure SmoothResize( var Src, Dst : TBitmap)
7747:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7748:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7749:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7750:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7751:     Procedure GrayScale( const Dst : TBitmap)
7752:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7753:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7754:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7755:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7756:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7757:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7758:     Procedure AntiAlias( const Dst : TBitmap)
7759:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7760:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7761:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7762:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7763:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7764:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7765:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
```

```
7766:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7767:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7768:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7769:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7770:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7771:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7772:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7773:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7774:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7775:     Procedure Invert( Src : TBitmap)
7776:     Procedure MirrorRight( Src : TBitmap)
7777:     Procedure MirrorDown( Src : TBitmap)
7778:   end;
7779: end;
7780:
7781: (*----------------------------------------------------------------------------*)
7782: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7783: begin
7784:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7785:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7786:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7787:   SIRegister_TJvPaintFX(CL);
7788:  Function SplineFilter( Value : Single) : Single
7789:  Function BellFilter( Value : Single) : Single
7790:  Function TriangleFilter( Value : Single) : Single
7791:  Function BoxFilter( Value : Single) : Single
7792:  Function HermiteFilter( Value : Single) : Single
7793:  Function Lanczos3Filter( Value : Single) : Single
7794:  Function MitchellFilter( Value : Single) : Single
7795: end;
7796:
7797:
7798: (*----------------------------------------------------------------------------*)
7799: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7800: begin
7801:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7802:  TeeMsg_DefaultSeriesName','String 'Series
7803:  TeeMsg_DefaultToolName','String 'ChartTool
7804:  ChartComponentPalette','String 'TeeChart
7805:  TeeMaxLegendColumns',LongInt'( 2);
7806:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7807:  TeeTitleFootDistance,LongInt( 5);
7808:   SIRegister_TCustomChartWall(CL);
7809:   SIRegister_TChartWall(CL);
7810:   SIRegister_TChartLegendGradient(CL);
7811:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7812:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7813:   FindClass('TOBJECT'),'LegendException
7814:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7815:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7816:   FindClass('TOBJECT'),'TCustomChartLegend
7817:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7818:   TLegendSymbolPosition', '( spLeft, spRight )
7819:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7820:   TSymbolCalcHeight', 'Function  : Integer
7821:   SIRegister_TLegendSymbol(CL);
7822:   SIRegister_TTeeCustomShapePosition(CL);
7823:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7824:   SIRegister_TLegendTitle(CL);
7825:   SIRegister_TLegendItem(CL);
7826:   SIRegister_TLegendItems(CL);
7827:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7828:   FindClass('TOBJECT'),'TCustomChart
7829:   SIRegister_TCustomChartLegend(CL);
7830:   SIRegister_TChartLegend(CL);
7831:   SIRegister_TChartTitle(CL);
7832:   SIRegister_TChartFootTitle(CL);
7833:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7834:    +'eButton; Shift : TShiftState; X, Y : Integer)
7835:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7836:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7837:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7838:    +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7839:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7840:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7841:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7842:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7843:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7844:    +'toMax : Boolean; Min : Double; Max : Double; end
7845:   TAllAxisSavedScales', 'array of TAxisSavedScales
7846:   SIRegister_TChartBackWall(CL);
7847:   SIRegister_TChartRightWall(CL);
7848:   SIRegister_TChartBottomWall(CL);
7849:   SIRegister_TChartLeftWall(CL);
7850:   SIRegister_TChartWalls(CL);
7851:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7852:   SIRegister_TCustomChart(CL);
7853:   SIRegister_TChart(CL);
7854:   SIRegister_TTeeSeriesTypes(CL);
```

```
7855:   SIRegister_TTeeToolTypes(CL);
7856:   SIRegister_TTeeDragObject(CL);
7857:   SIRegister_TColorPalettes(CL);
7858:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
       AGalleryPage:PString;ANumGallerySeries:Integer;
7859:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7860:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
       AGalleryPage:PString;ANumGallerySeries: Int;
7861:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7862:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
       ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7863:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7864:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7865:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7866:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7867:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
       AFunctionClass : TTeeFunctionClass) : TChartSeries
7868:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7869:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7870:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7871:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7872:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7873:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7874:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7875:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7876:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7877:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7878:  Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
       R:TRect;ReferenceChart:TCustomChart);
7879:   SIRegister_TChartTheme(CL);
7880:   //TChartThemeClass', 'class of TChartTheme
7881:   //TCanvasClass', 'class of TCanvas3D
7882:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7883:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7884:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7885:  Procedure ShowMessageUser( const S : String)
7886:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7887:  Function HasLabels( ASeries : TChartSeries) : Boolean
7888:  Function HasColors( ASeries : TChartSeries) : Boolean
7889:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7890:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7891: end;
7892:
7893:
7894: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7895: begin
7896:  //'TeeFormBorderStyle',' bsNone);
7897:   SIRegister_TMetafile(CL);
7898:  'TeeDefVerticalMargin','LongInt'( 4);
7899:  'TeeDefHorizMargin','LongInt'( 3);
7900:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7901:  'TeeMsg_TeeHand','String 'crTeeHand
7902:  'TeeNormalPrintDetail','LongInt'( 0);
7903:  'TeeHighPrintDetail','LongInt'( - 100);
7904:  'TeeDefault_PrintMargin','LongInt'( 15);
7905:  'MaxDefaultColors','LongInt'( 19);
7906:  'TeeTabDelimiter','Char #9);
7907:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7908:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7909:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7910:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7911:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7912:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7913:   SIRegister_TCustomPanelNoCaption(CL);
7914:   FindClass('TOBJECT'),'TCustomTeePanel
7915:   SIRegister_TZoomPanning(CL);
7916:   SIRegister_TTeeEvent(CL);
7917:   //SIRegister_TTeeEventListeners(CL);
7918:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7919:   SIRegister_TTeeMouseEvent(CL);
7920:   SIRegister_TCustomTeePanel(CL);
7921:   //TChartGradient', 'TTeeGradient
7922:   //TChartGradientClass', 'class of TChartGradient
7923:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7924:   SIRegister_TTeeZoomPen(CL);
7925:   SIRegister_TTeeZoomBrush(CL);
7926:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7927:   SIRegister_TTeeZoom(CL);
7928:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7929:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7930:   SIRegister_TBackImage(CL);
7931:   SIRegister_TCustomTeePanelExtended(CL);
7932:   //TChartBrushClass', 'class of TChartBrush
7933:   SIRegister_TTeeCustomShapeBrushPen(CL);
7934:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7935:   TTextFormat', '( ttfNormal, ttfHtml )
7936:   SIRegister_TTeeCustomShape(CL);
7937:   SIRegister_TTeeShape(CL);
7938:   SIRegister_TTeeExportData(CL);
```

```
7939:  Function TeeStr( const Num : Integer) : String
7940:  Function DateTimeDefaultFormat( const AStep : Double) : String
7941:  Function TEEDaysInMonth( Year, Month : Word) : Word
7942:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7943:  Function NextDateTimeStep( const AStep : Double) : Double
7944:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7945:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7946:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7947:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7948:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7949:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7950:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7951:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7952:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7953:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7954:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7955:  Function DelphiToLocalFormat( const Format : String) : String
7956:  Function LocalToDelphiFormat( const Format : String) : String
7957:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
7958:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7959:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7960:    TTeeSortCompare', 'Function ( a, b : Integer) : Integer
7961:    TTeeSortSwap', 'Procedure ( a, b : Integer)
7962:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
7963:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
7964:  Function TeeExtractField( St : String; Index : Integer) : String;
7965:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7966:  Function TeeNumFields( St : String) : Integer;
7967:  Function TeeNumFields1( const St, Separator : String) : Integer;
7968:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7969:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7970:  // TColorArray', 'array of TColor
7971:  Function GetDefaultColor( const Index : Integer) : TColor
7972:  Procedure SetDefaultColorPalette;
7973:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
7974:  'TeeCheckBoxSize','LongInt'( 11);
7975:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
7976:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
7977:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
7978:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
7979:  Procedure TeeTranslateControl( AControl : TControl);
7980:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
7981:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
7982:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
7983:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
7984:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
7985:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
7986:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
7987:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
7988:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
7989:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
7990:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
7991:  Procedure TeeSaveStringOption( const AKey, Value : String)
7992:  Function TeeDefaultXMLEncoding : String
7993:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
7994:   TeeWindowHandle', 'Integer
7995:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
7996:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
7997:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
7998: end;
7999:
8000:
8001: using mXBDEUtils
8002: ********************************************************************************
8003:  Procedure SetAlias( aAlias, aDirectory : String)
8004:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
8005:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
8006:  Procedure SetBDE( aPath, aNode, aValue : String)
8007:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8008:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8009:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8010:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8011:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8012:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8013:
8014:
8015: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
8016: begin
8017: AddClassN(FindClass('TOBJECT'),'EDateTime
8018: Function DatePart( const D : TDateTime) : Integer
8019: Function TimePart( const D : TDateTime) : Double
8020: Function Century( const D : TDateTime) : Word
8021: Function Year( const D : TDateTime) : Word
8022: Function Month( const D : TDateTime) : Word
8023: Function Day( const D : TDateTime) : Word
8024: Function Hour( const D : TDateTime) : Word
8025: Function Minute( const D : TDateTime) : Word
```

```
8026: Function Second( const D : TDateTime) : Word
8027: Function Millisecond( const D : TDateTime) : Word
8028: ('OneDay','Extended').setExtended( 1.0);
8029: ('OneHour','Extended').SetExtended( OneDay / 24);
8030: ('OneMinute','Extended').SetExtended( OneHour / 60);
8031: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8032: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8033: ('OneWeek','Extended').SetExtended( OneDay * 7);
8034: ('HoursPerDay','Extended').SetExtended( 24);
8035: ('MinutesPerHour','Extended').SetExtended( 60);
8036: ('SecondsPerMinute','Extended').SetExtended( 60);
8037: Procedure SetYear( var D : TDateTime; const Year : Word)
8038: Procedure SetMonth( var D : TDateTime; const Month : Word)
8039: Procedure SetDay( var D : TDateTime; const Day : Word)
8040: Procedure SetHour( var D : TDateTime; const Hour : Word)
8041: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8042: Procedure SetSecond( var D : TDateTime; const Second : Word)
8043: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8044: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8045: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8046: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8047: Function IsAM( const D : TDateTime) : Boolean
8048: Function IsPM( const D : TDateTime) : Boolean
8049: Function IsMidnight( const D : TDateTime) : Boolean
8050: Function IsNoon( const D : TDateTime) : Boolean
8051: Function IsSunday( const D : TDateTime) : Boolean
8052: Function IsMonday( const D : TDateTime) : Boolean
8053: Function IsTuesday( const D : TDateTime) : Boolean
8054: Function IsWedneday( const D : TDateTime) : Boolean
8055: Function IsThursday( const D : TDateTime) : Boolean
8056: Function IsFriday( const D : TDateTime) : Boolean
8057: Function IsSaturday( const D : TDateTime) : Boolean
8058: Function IsWeekend( const D : TDateTime) : Boolean
8059: Function Noon( const D : TDateTime) : TDateTime
8060: Function Midnight( const D : TDateTime) : TDateTime
8061: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8062: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8063: Function NextWorkday( const D : TDateTime) : TDateTime
8064: Function PreviousWorkday( const D : TDateTime) : TDateTime
8065: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8066: Function LastDayOfYear( const D : TDateTime) : TDateTime
8067: Function EasterSunday( const Year : Word) : TDateTime
8068: Function GoodFriday( const Year : Word) : TDateTime
8069: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8070: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8071: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8072: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8073: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8074: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8075: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8076: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8077: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8078: Function DayOfYear( const D : TDateTime) : Integer
8079: Function DaysInMonth( const Ye, Mo : Word) : Integer
8080: Function DaysInMonth( const D : TDateTime) : Integer
8081: Function DaysInYear( const Ye : Word) : Integer
8082: Function DaysInYearDate( const D : TDateTime) : Integer
8083: Function WeekNumber( const D : TDateTime) : Integer
8084: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8085: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8086: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8087: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8088: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8089: Function DiffHours( const D1, D2 : TDateTime) : Integer
8090: Function DiffDays( const D1, D2 : TDateTime) : Integer
8091: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8092: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8093: Function DiffYears( const D1, D2 : TDateTime) : Integer
8094: Function GMTBias : Integer
8095: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8096: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8097: Function NowAsGMTTime : TDateTime
8098: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8099: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8100: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8101: Function DateTimeToANSI( const D : TDateTime) : Integer
8102: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8103: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8104: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8105: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8106: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8107: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8108: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8109: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8110: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8111: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8112: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8113: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8114: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
```

```
8115: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8116: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8117: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8118: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8119: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8120: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8121: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8122: Function EnglishShortMonthA( const S : AnsiString) : Integer
8123: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8124: Function EnglishLongMonthA( const S : AnsiString) : Integer
8125: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8126: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8127: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8128: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8129: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8130: Function RFCMonthA( const S : AnsiString) : Word
8131: Function RFCMonthU( const S : UnicodeString) : Word
8132: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
8133: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8134: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8135: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8136: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8137: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8138: Function NowAsRFCDateTimeA : AnsiString
8139: Function NowAsRFCDateTimeU : UnicodeString
8140: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8141: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8142: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8143: Function TimePeriodStr( const D : TDateTime) : AnsiString
8144: Procedure SelfTest
8145: end;
8146: //*****************************************CFileUtils
8147: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8148: Function PathHasDriveLetter( const Path : String) : Boolean
8149: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8150: Function PathIsDriveLetter( const Path : String) : Boolean
8151: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8152: Function PathIsDriveRoot( const Path : String) : Boolean
8153: Function PathIsRootA( const Path : AnsiString) : Boolean
8154: Function PathIsRoot( const Path : String) : Boolean
8155: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8156: Function PathIsUNCPath( const Path : String) : Boolean
8157: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8158: Function PathIsAbsolute( const Path : String) : Boolean
8159: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8160: Function PathIsDirectory( const Path : String) : Boolean
8161: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8162: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8163: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8164: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8165: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8166: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8167: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8168: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8169: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8170: Function PathCanonical( const Path : String; const PathSep : Char) : String
8171: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8172: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8173: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8174: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8175: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8176: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8177: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8178: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8179: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8180: Function FileNameValid( const FileName : String) : String
8181: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8182: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8183: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8184: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8185: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8186: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8187: Procedure CCopyFile( const FileName, DestName : String)
8188: Procedure CMoveFile( const FileName, DestName : String)
8189: Function CDeleteFiles( const FileMask : String) : Boolean
8190: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8191: Procedure FileCloseEx( const FileHandle : TFileHandle)
8192: Function FileExistsA( const FileName : AnsiString) : Boolean
8193:  Function CFileExists( const FileName : String) : Boolean
8194:  Function CFileGetSize( const FileName : String) : Int64
8195: Function FileGetDateTime( const FileName : String) : TDateTime
8196: Function FileGetDateTime2( const FileName : String) : TDateTime
8197: Function FileIsReadOnly( const FileName : String) : Boolean
8198: Procedure FileDeleteEx( const FileName : String)
8199: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8200: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
      : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8201: Function DirectoryEntryExists( const Name : String) : Boolean
8202: Function DirectoryEntrySize( const Name : String) : Int64
```

```
8203:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8204: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8205: Procedure CDirectoryCreate( const DirectoryName : String)
8206: Function GetFirstFileNameMatching( const FileMask : String) : String
8207: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8208: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8209: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8210:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8211:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8212: Function DriveIsValid( const Drive : Char) : Boolean
8213: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8214: Function DriveFreeSpace( const Path : AnsiString) : Int64
8215:
8216: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8217: begin
8218:  AddClassN(FindClass('TOBJECT'),'ETimers
8219:  Const('TickFrequency','LongInt'( 1000);Function GetTick : LongWord
8220: Function TickDelta( const D1, D2 : LongWord) : Integer
8221: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8222:   AddTypeS('THPTimer', 'Int64
8223: Procedure StartTimer( var Timer : THPTimer)
8224: Procedure StopTimer( var Timer : THPTimer)
8225: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8226: Procedure InitStoppedTimer( var Timer : THPTimer)
8227: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8228: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8229: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8230: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8231: Function GetHighPrecisionFrequency : Int64
8232: Function GetHighPrecisionTimerOverhead : Int64
8233: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8234: Procedure SelfTestCTimer
8235: end;
8236:
8237: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8238: begin
8239:  Function RandomSeed : LongWord
8240:  Procedure AddEntropy( const Value : LongWord)
8241:  Function RandomUniform : LongWord;
8242:  Function RandomUniform1( const N : Integer) : Integer;
8243:  Function RandomBoolean : Boolean
8244:  Function RandomByte : Byte
8245:  Function RandomByteNonZero : Byte
8246:  Function RandomWord : Word
8247:  Function RandomInt64 : Int64;
8248:  Function RandomInt641( const N : Int64) : Int64;
8249:  Function RandomHex( const Digits : Integer) : String
8250:  Function RandomFloat : Extended
8251:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8252:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8253:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8254:  Function mwcRandomLongWord : LongWord
8255:  Function urnRandomLongWord : LongWord
8256:  Function moaRandomFloat : Extended
8257:  Function mwcRandomFloat : Extended
8258:  Function RandomNormalF : Extended
8259:  Procedure SelfTestCRandom
8260: end;
8261:
8262: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8263: begin
8264:  // PIntArray', '^TIntArray // will not work
8265:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8266:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8267:  Function synMax( x, y : integer) : integer
8268:  Function synMin( x, y : integer) : integer
8269:  Function synMinMax( x, mi, ma : integer) : integer
8270:  Procedure synSwapInt( var l, r : integer)
8271:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8272:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8273:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8274:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8275:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8276:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8277:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8278:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8279:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8280:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8281:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8282:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8283:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8284:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8285:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8286:  ('C3_NONSPACING','LongInt'( 1);
8287:  'C3_DIACRITIC','LongInt'( 2);
8288:  'C3_VOWELMARK','LongInt'( 4);
8289:  ('C3_SYMBOL','LongInt'( 8);
8290:  ('C3_KATAKANA','LongWord( $0010);
8291:  ('C3_HIRAGANA','LongWord( $0020);
```

```
8292:  ('C3_HALFWIDTH','LongWord( $0040);
8293:  ('C3_FULLWIDTH','LongWord( $0080);
8294:  ('C3_IDEOGRAPH','LongWord( $0100);
8295:  ('C3_KASHIDA','LongWord( $0200);
8296:  ('C3_LEXICAL','LongWord( $0400);
8297:  ('C3_ALPHA','LongWord( $8000);
8298:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8299:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8300:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8301:  Function synIsStringType( Value : Word) : TStringType
8302:  Function synGetEOL( Line : PChar) : PChar
8303:  Function synEncodeString( s : string) : string
8304:  Function synDecodeString( s : string) : string
8305:  Procedure synFreeAndNil( var Obj: TObject)
8306:  Procedure synAssert( Expr : Boolean)
8307:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8308:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8309:   TReplaceFlags', 'set of TReplaceFlag )
8310:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8311:  Function synGetRValue( RGBValue : TColor) : byte
8312:  Function synGetGValue( RGBValue : TColor) : byte
8313:  Function synGetBValue( RGBValue : TColor) : byte
8314:  Function synRGB( r, g, b : Byte) : Cardinal
8315: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8316:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8317:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8318:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8319:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8320: end;
8321:
8322:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8323:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8324:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8325:
8326: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8327: begin
8328:  Function STimeZoneBias : integer
8329:  Function TimeZone : string
8330:  Function Rfc822DateTime( t : TDateTime) : string
8331:  Function CDateTime( t : TDateTime) : string
8332:  Function SimpleDateTime( t : TDateTime) : string
8333:  Function AnsiCDateTime( t : TDateTime) : string
8334:  Function GetMonthNumber( Value : String) : integer
8335:  Function GetTimeFromStr( Value : string) : TDateTime
8336:  Function GetDateMDYFromStr( Value : string) : TDateTime
8337:  Function DecodeRfcDateTime( Value : string) : TDateTime
8338:  Function GetUTTime : TDateTime
8339:  Function SetUTTime( Newdt : TDateTime) : Boolean
8340:  Function SGetTick : LongWord
8341:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8342:  Function CodeInt( Value : Word) : Ansistring
8343:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8344:  Function CodeLongInt( Value : LongInt) : Ansistring
8345:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8346:  Function DumpStr( const Buffer : Ansistring) : string
8347:  Function DumpExStr( const Buffer : Ansistring) : string
8348:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8349:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8350:  Function TrimSPLeft( const S : string) : string
8351:  Function TrimSPRight( const S : string) : string
8352:  Function TrimSP( const S : string) : string
8353:  Function SeparateLeft( const Value, Delimiter : string) : string
8354:  Function SeparateRight( const Value, Delimiter : string) : string
8355:  Function SGetParameter( const Value, Parameter : string) : string
8356:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8357:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8358:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8359:  Function GetEmailAddr( const Value : string) : string
8360:  Function GetEmailDesc( Value : string) : string
8361:  Function CStrToHex( const Value : Ansistring) : string
8362:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8363:  Function CBinToInt( const Value : string) : Integer
8364:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8365:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8366:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8367:  Function CRPos( const Sub, Value : String) : Integer
8368:  Function FetchBin( var Value : string; const Delimiter : string) : string
8369:  Function CFetch( var Value : string; const Delimiter : string) : string
8370:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8371:  Function IsBinaryString( const Value : AnsiString) : Boolean
8372:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8373:  Procedure StringsTrim( const value : TStrings)
8374:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8375:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8376:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8377:  Function CCountOfChar( const Value : string; aChr : char) : integer
8378:  Function UnquoteStr( const Value : string; Quote : Char) : string
```

```
8379:  Function QuoteStr( const Value : string; Quote : Char) : string
8380:  Procedure HeadersToList( const Value : TStrings)
8381:  Procedure ListToHeaders( const Value : TStrings)
8382:  Function SwapBytes( Value : integer) : integer
8383:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8384:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8385:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8386:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8387:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8388:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8389: end;
8390:
8391: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8392: begin
8393:  ('CrcBufSize','LongInt'( 2048);
8394:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8395:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8396:  Function Adler32OfFile( FileName : AnsiString) : LongInt
8397:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8398:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8399:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8400:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8401:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8402:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8403:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8404:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8405:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8406:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8407:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8408:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8409: end;
8410:
8411: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8412: begin
8413:  function CreateOleObject(const ClassName: String): IDispatch;
8414:  function GetActiveOleObject(const ClassName: String): IDispatch;
8415:  function ProgIDToClassID(const ProgID: string): TGUID;
8416:  function ClassIDToProgID(const ClassID: TGUID): string;
8417:  function CreateClassID: string;
8418:  function CreateGUIDString: string;
8419:  function CreateGUIDID: string;
8420:  procedure OleError(ErrorCode: longint)
8421:  procedure OleCheck(Result: HResult);
8422: end;
8423:
8424:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8425:  Function xGetActiveOleObject( const ClassName : string) : Variant
8426:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8427:  Function DllCanUnloadNow : HResult
8428:  Function DllRegisterServer : HResult
8429:  Function DllUnregisterServer : HResult
8430:  Function VarFromInterface( Unknown : IUnknown) : Variant
8431:  Function VarToInterface( const V : Variant) : IDispatch
8432:  Function VarToAutoObject( const V : Variant) : TAutoObject
8433:  //Procedure
      DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8434:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8435:  Procedure OleError( ErrorCode : HResult)
8436:  Procedure OleCheck( Result : HResult)
8437:  Function StringToClassID( const S : string) : TCLSID
8438:  Function ClassIDToString( const ClassID : TCLSID) : string
8439:  Function xProgIDToClassID( const ProgID : string) : TCLSID
8440:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8441:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8442:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8443:  Function xGUIDToString( const ClassID : TGUID) : string
8444:  Function xStringToGUID( const S : string) : TGUID
8445:  Function xGetModuleName( Module : HMODULE) : string
8446:  Function xAcquireExceptionObject : TObject
8447:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8448:  Function xUtf8Encode( const WS : WideString) : UTF8String
8449:  Function xUtf8Decode( const S : UTF8String) : WideString
8450:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8451:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8452:  Function XRTLHandleCOMException : HResult
8453:  Procedure XRTLCheckArgument( Flag : Boolean)
8454:  //Procedure XRTLCheckOutArgument( out Arg)
8455:  Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8456:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8457:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8458:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8459:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8460:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8461: function   XRTLDefaultCategoryManager: IUnknown;
8462: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8463: // ICatRegister helper functions
8464: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
```

```
8465:                                     LocaleID: TLCID = LOCALE_USER_DEFAULT;
8466:                                     const CategoryManager: IUnknown = nil): HResult;
8467: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8468:                                     LocaleID: TLCID = LOCALE_USER_DEFAULT;
8469:                                     const CategoryManager: IUnknown = nil): HResult;
8470: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8471:                                     const CategoryManager: IUnknown = nil): HResult;
8472: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8473:                                       const CategoryManager: IUnknown = nil): HResult;
8474: // ICatInformation helper functions
8475: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8476:                                     LocaleID: TLCID = LOCALE_USER_DEFAULT;
8477:                                     const CategoryManager: IUnknown = nil): HResult;
8478: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8479:                                const CategoryManager: IUnknown = nil): HResult;
8480: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8481:                                   const CategoryManager: IUnknown = nil): HResult;
8482: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8483:                                   const CategoryManager: IUnknown = nil): HResult;
8484: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8485:                    const ADelete: Boolean = True): WideString;
8486: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8487: Function XRTLGetVariantAsString( const Value : Variant) : string
8488: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8489: Function XRTLGetTimeZones : TXRTLTimeZones
8490: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8491: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8492: Function GMTNow : TDateTime
8493: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8494: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8495: Procedure XRTLNotImplemented
8496: Procedure XRTLRaiseError( E : Exception)
8497: Procedure XRTLRaise( E : Exception)');
8498: Procedure XRaise( E : Exception)');
8499: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8500:
8501:
8502: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8503: begin
8504:   SIRegister_IXRTLValue(CL);
8505:   SIRegister_TXRTLValue(CL);
8506:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8507:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8508: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8509: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8510: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8511: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8512: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8513: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8514: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8515: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8516: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8517: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8518: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8519: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8520: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8521: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8522: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8523: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8524: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8525: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8526: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8527: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8528: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8529: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8530: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8531: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8532: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8533: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8534: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8535:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8536: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8537: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8538: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8539: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8540: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8541: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8542: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8543: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8544: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8545: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8546: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8547: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8548: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8549: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8550: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8551: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8552: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
```

```
8553: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8554: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8555: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8556: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8557: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8558: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8559: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8560: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8561: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8562: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8563: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8564: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8565: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8566: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8567: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8568: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8569: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8570: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8571: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8572: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8573: end;
8574:
8575: //*****************************unit uPSI_GR32;****************************************
8576:
8577:  Function Color32( WinColor : TColor) : TColor32;
8578:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8579:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8580:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8581:  Function WinColor( Color32 : TColor32) : TColor
8582:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8583:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8584:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8585:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8586:  Function RedComponent( Color32 : TColor32) : Integer
8587:  Function GreenComponent( Color32 : TColor32) : Integer
8588:  Function BlueComponent( Color32 : TColor32) : Integer
8589:  Function AlphaComponent( Color32 : TColor32) : Integer
8590:  Function Intensity( Color32 : TColor32) : Integer
8591:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8592:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8593:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8594:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8595:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8596:  Function WinPalette( const P : TPalette32) : HPALETTE
8597:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8598:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8599:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8600:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8601:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8602:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8603:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8604:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8605:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8606:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8607:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8608:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8609:  Function FixedRect1( const ARect : TRect) : TRect;
8610:  Function FixedRect2( const FR : TFloatRect) : TRect;
8611:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8612:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8613:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8614:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8615:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8616:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8617:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8618:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8619:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8620:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8621:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8622:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8623:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8624:  Function IsRectEmpty( const R : TRect) : Boolean;
8625:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8626:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8627:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8628:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8629:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8630:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8631:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8632:  Function MessageBeep( uType : UINT) : BOOL
8633:  Function ShowCursor( bShow : BOOL) : Integer
8634:  Function SetCursorPos( X, Y : Integer) : BOOL
8635:  Function SetCursor( hCursor : HICON) : HCURSOR
8636:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8637:  //Function ClipCursor( lpRect : PRect) : BOOL
8638:  Function GetClipCursor( var lpRect : TRect) : BOOL
8639:  Function GetCursor : HCURSOR
8640:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8641:  Function GetCaretBlinkTime : UINT
```

```
8642:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8643:  Function DestroyCaret : BOOL
8644:  Function HideCaret( hWnd : HWND) : BOOL
8645:  Function ShowCaret( hWnd : HWND) : BOOL
8646:  Function SetCaretPos( X, Y : Integer) : BOOL
8647:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8648:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8649:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8650:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8651:  Function WindowFromPoint( Point : TPoint) : HWND
8652:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8653:
8654:
8655: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8656: begin
8657:  Function FixedFloor( A : TFixed) : Integer
8658:  Function FixedCeil( A : TFixed) : Integer
8659:  Function FixedMul( A, B : TFixed) : TFixed
8660:  Function FixedDiv( A, B : TFixed) : TFixed
8661:  Function OneOver( Value : TFixed) : TFixed
8662:  Function FixedRound( A : TFixed) : Integer
8663:  Function FixedSqr( Value : TFixed) : TFixed
8664:  Function FixedSqrtLP( Value : TFixed) : TFixed
8665:  Function FixedSqrtHP( Value : TFixed) : TFixed
8666:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8667:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8668:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8669:  Function GRHypot( const X, Y : TFloat) : TFloat;
8670:  Function Hypot1( const X, Y : Integer) : Integer;
8671:  Function FastSqrt( const Value : TFloat) : TFloat
8672:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8673:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8674:  Function FastInvSqrt( const Value : Single) : Single;
8675:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8676:  Function GRIsPowerOf2( Value : Integer) : Boolean
8677:  Function PrevPowerOf2( Value : Integer) : Integer
8678:  Function NextPowerOf2( Value : Integer) : Integer
8679:  Function Average( A, B : Integer) : Integer
8680:  Function GRSign( Value : Integer) : Integer
8681:  Function FloatMod( x, y : Double) : Double
8682: end;
8683:
8684: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8685: begin
8686:  Function Clamp( const Value : Integer) : Integer;
8687:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8688:  Function StackAlloc( Size : Integer) : Pointer
8689:  Procedure StackFree( P : Pointer)
8690:  Procedure Swap( var A, B : Pointer);
8691:  Procedure Swap1( var A, B : Integer);
8692:  Procedure Swap2( var A, B : TFixed);
8693:  Procedure Swap3( var A, B : TColor32);
8694:  Procedure TestSwap( var A, B : Integer);
8695:  Procedure TestSwap1( var A, B : TFixed);
8696:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8697:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8698:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8699:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8700:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8701:  Function GRMin( const A, B, C : Integer) : Integer;
8702:  Function GRMax( const A, B, C : Integer) : Integer;
8703:  Function Clamp( Value, Max : Integer) : Integer;
8704:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8705:  Function Wrap( Value, Max : Integer) : Integer;
8706:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8707:  Function Wrap3( Value, Max : Single) : Single;;
8708:  Function WrapPow2( Value, Max : Integer) : Integer;
8709:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8710:  Function Mirror( Value, Max : Integer) : Integer;
8711:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8712:  Function MirrorPow2( Value, Max : Integer) : Integer;
8713:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8714:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8715:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8716:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8717:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8718:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8719:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8720:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8721:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8722:  Function Div255( Value : Cardinal) : Cardinal
8723:  Function SAR_4( Value : Integer) : Integer
8724:  Function SAR_8( Value : Integer) : Integer
8725:  Function SAR_9( Value : Integer) : Integer
8726:  Function SAR_11( Value : Integer) : Integer
8727:  Function SAR_12( Value : Integer) : Integer
8728:  Function SAR_13( Value : Integer) : Integer
8729:  Function SAR_14( Value : Integer) : Integer
8730:  Function SAR_15( Value : Integer) : Integer
```

```
8731:  Function SAR_16( Value : Integer) : Integer
8732:  Function ColorSwap( WinColor : TColor) : TColor32
8733: end;
8734:
8735: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8736: begin
8737:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8738:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8739:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8740:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8741:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8742:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8743:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8744:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8745:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8746:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8747:  Function CreateBitmask( Components : TColor32Components) : TColor32
8748:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8749:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8750:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8751: end;
8752:
8753:
8754: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8755: begin
8756:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8757:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8758:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8759:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8760:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8761:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8762:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8763:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8764:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8765:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8766:    //+'tedRangeBuffer; MoreData : Boolean; end
8767:  Function NtfsSetSparse( const FileName : string) : Boolean
8768:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8769:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8770:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8771:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8772:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8773:  Function NtfsGetSparse( const FileName : string) : Boolean
8774:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8775:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8776:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8777:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8778:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8779:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8780:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8781:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8782:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8783:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8784:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8785:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8786:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8787:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8788:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8789:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8790:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8791:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8792:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8793:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8794:   AddTypeS('TStreamIds', 'set of TStreamId
8795:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8796:    +': ___Pointer; StreamIds : TStreamIds; end
8797:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8798:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8799:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8800:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8801:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8802:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8803:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8804:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8805:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
      List:TStrings):Bool;
8806:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8807:  Function JclAppInstances : TJclAppInstances;
8808:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8809:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8810:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8811:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8812:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8813:
```

```
8814:
8815: (*----------------------------------------------------------------------------*)
8816: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8817: begin
8818:   FindClass('TOBJECT'),'EJclGraphicsError
8819:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8820:   TDynPointArray', 'array of TPoint
8821:   TDynDynPointArrayArray', 'array of TDynPointArray
8822:   TPointF', 'record X : Single; Y : Single; end
8823:   TDynPointArrayF', 'array of TPointF
8824:   TDrawMode2', '( dmOpaque, dmBlend )
8825:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8826:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8827:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8828:   TMatrix3d', 'record array[0..2,0..2] of extended end
8829:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8830:   TScanLine', 'array of Integer
8831:   TScanLines', 'array of TScanLine
8832:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8833:   TGradientDirection', '( gdVertical, gdHorizontal )
8834:   TPolyFillMode', '( fmAlternate, fmWinding )
8835:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8836:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8837:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8838:   SIRegister_TJclDesktopCanvas(CL);
8839:   FindClass('TOBJECT'),'TJclRegion
8840:   SIRegister_TJclRegionInfo(CL);
8841:   SIRegister_TJclRegion(CL);
8842:   SIRegister_TJclThreadPersistent(CL);
8843:   SIRegister_TJclCustomMap(CL);
8844:   SIRegister_TJclBitmap32(CL);
8845:   SIRegister_TJclByteMap(CL);
8846:   SIRegister_TJclTransformation(CL);
8847:   SIRegister_TJclLinearTransformation(CL);
8848:  Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8849:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8850:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8851:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8852:  Procedure BitmapToJPeg( const FileName : string)
8853:  Procedure JPegToBitmap( const FileName : string)
8854:  Function ExtractIconCount( const FileName : string) : Integer
8855:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8856:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8857:  Procedure
      BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8858:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8859:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8860:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8861:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
      TGradientDirection) : Boolean;
8862:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8863:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8864:  Procedure ScreenShot1( bm : TBitmap);
8865:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8866:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8867:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8868:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8869:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8870:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8871:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8872:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8873:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8874:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8875:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8876:  Procedure Invert( Dst, Src : TJclBitmap32)
8877:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8878:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8879:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8880:  Procedure SetGamma( Gamma : Single)
8881: end;
8882:
8883: (*----------------------------------------------------------------------------*)
8884: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8885: begin
8886:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8887:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8888:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8889:  Function LockedDec( var Target : Integer) : Integer
8890:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8891:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8892:  Function LockedExchangeDec( var Target : Integer) : Integer
8893:  Function LockedExchangeInc( var Target : Integer) : Integer
8894:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8895:  Function LockedInc( var Target : Integer) : Integer
8896:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8897:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
```

```
8898:   SIRegister_TJclDispatcherObject(CL);
8899:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8900:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8901:   SIRegister_TJclCriticalSection(CL);
8902:   SIRegister_TJclCriticalSectionEx(CL);
8903:   SIRegister_TJclEvent(CL);
8904:   SIRegister_TJclWaitableTimer(CL);
8905:   SIRegister_TJclSemaphore(CL);
8906:   SIRegister_TJclMutex(CL);
8907:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8908:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8909:   SIRegister_TJclOptex(CL);
8910:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8911:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8912:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8913:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8914:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8915:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8916:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8917:   PMeteredSection', '^TMeteredSection // will not work
8918:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8919:   SIRegister_TJclMeteredSection(CL);
8920:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8921:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8922:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8923:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8924:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8925:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8926:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8927:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8928:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8929:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8930:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8931:   FindClass('TOBJECT'),'EJclCriticalSectionError
8932:   FindClass('TOBJECT'),'EJclEventError
8933:   FindClass('TOBJECT'),'EJclWaitableTimerError
8934:   FindClass('TOBJECT'),'EJclSemaphoreError
8935:   FindClass('TOBJECT'),'EJclMutexError
8936:   FindClass('TOBJECT'),'EJclMeteredSectionError
8937: end;
8938:
8939:
8940: //*************************unit uPSI_mORMotReport;
8941: Procedure SetCurrentPrinterAsDefault
8942: Function CurrentPrinterName : string
8943: Function mCurrentPrinterPaperSize : string
8944: Procedure UseDefaultPrinter
8945:
8946: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
8947: begin
8948:   with FindClass('TOBJECT'), 'TStream') do begin
8949:     IsAbstract := True;
8950:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8951:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8952:     function Read(Buffer:String;Count:LongInt):LongInt
8953:     function Write(Buffer:String;Count:LongInt):LongInt
8954:     function ReadString(Buffer:String;Count:LongInt):LongInt    //FileStream
8955:     function WriteString(Buffer:String;Count:LongInt):LongInt
8956:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
8957:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
8958:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8959:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8960:
8961:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8962:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8963:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8964:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8965:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8966:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8967:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8968:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8969:
8970:     function Seek(Offset:LongInt;Origin:Word):LongInt
8971:     procedure ReadBuffer(Buffer:String;Count:LongInt)
8972:     procedure WriteBuffer(Buffer:String;Count:LongInt)
8973:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
8974:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)');
8975:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)');
8976:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)');
8977:
8978:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
8979:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
8980:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
8981:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
8982:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8983:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8984:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
```

```
8985:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8986:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8987:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8988:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
8989:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
8990:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
8991:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
8992:
8993:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
8994:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
8995:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
8996:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)');
8997:     //READBUFFERAC
8998:     function InstanceSize: Longint
8999:     Procedure FixupResourceHeader( FixupInfo : Integer)
9000:     Procedure ReadResHeader
9001:
9002:     {$IFDEF DELPHI4UP}
9003:     function CopyFrom(Source:TStream;Count:Int64):LongInt
9004:     {$ELSE}
9005:     function CopyFrom(Source:TStream;Count:Integer):LongInt
9006:     {$ENDIF}
9007:     RegisterProperty('Position', 'LongInt', iptrw);
9008:     RegisterProperty('Size', 'LongInt', iptrw);
9009:   end;
9010: end;
9011:
9012:
9013: { ****************************************************************
9014:   Unit DMATH - Interface for DMATH.DLL
9015:   **************************************************************** }
9016: // see more docs/dmath_manual.pdf
9017:
9018: Function InitEval : Integer
9019: Procedure SetVariable( VarName : Char; Value : Float)
9020: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9021: Function Eval( ExpressionString : String) : Float
9022:
9023: unit dmath; //types are in built, others are external in DLL
9024: interface
9025: {$IFDEF DELPHI}
9026: uses
9027:   StdCtrls, Graphics;
9028: {$ENDIF}
9029: { ----------------------------------------------------------------
9030:   Types and constants
9031:   ---------------------------------------------------------------- }
9032: {$i types.inc}
9033: { ----------------------------------------------------------------
9034:   Error handling
9035:   ---------------------------------------------------------------- }
9036: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9037: { Sets the error code }
9038: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9039: { Sets error code and default function value }
9040: function MathErr : Integer; external 'dmath';
9041: { Returns the error code }
9042: { ----------------------------------------------------------------
9043:   Dynamic arrays
9044:   ---------------------------------------------------------------- }
9045: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9046: { Sets the auto-initialization of arrays }
9047: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9048: { Creates floating point vector V[0..Ub] }
9049: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9050: { Creates integer vector V[0..Ub] }
9051: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9052: { Creates complex vector V[0..Ub] }
9053: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9054: { Creates boolean vector V[0..Ub] }
9055: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9056: { Creates string vector V[0..Ub] }
9057: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9058: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9059: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9060: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9061: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9062: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9063: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9064: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9065: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9066: { Creates string matrix A[0..Ub1, 0..Ub2] }
9067: { ----------------------------------------------------------------
9068:   Minimum, maximum, sign and exchange
9069:   ---------------------------------------------------------------- }
9070: function FMin(X, Y : Float) : Float; external 'dmath';
9071: { Minimum of 2 reals }
9072: function FMax(X, Y : Float) : Float; external 'dmath';
9073: { Maximum of 2 reals }
```

```
9074: function IMin(X, Y : Integer) : Integer; external 'dmath';
9075: { Minimum of 2 integers }
9076: function IMax(X, Y : Integer) : Integer; external 'dmath';
9077: { Maximum of 2 integers }
9078: function Sgn(X : Float) : Integer; external 'dmath';
9079: { Sign (returns 1 if X = 0) }
9080: function Sgn0(X : Float) : Integer; external 'dmath';
9081: { Sign (returns 0 if X = 0) }
9082: function DSgn(A, B : Float) : Float; external 'dmath';
9083: { Sgn(B) * |A| }
9084: procedure FSwap(var X, Y : Float); external 'dmath';
9085: { Exchange 2 reals }
9086: procedure ISwap(var X, Y : Integer); external 'dmath';
9087: { Exchange 2 integers }
9088: { ---------------------------------------------------------------
9089:   Rounding functions
9090:   --------------------------------------------------------------- }
9091: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9092: { Rounds X to N decimal places }
9093: function Ceil(X : Float) : Integer; external 'dmath';
9094: { Ceiling function }
9095: function Floor(X : Float) : Integer; external 'dmath';
9096: { Floor function }
9097: { ---------------------------------------------------------------
9098:   Logarithms, exponentials and power
9099:   --------------------------------------------------------------- }
9100: function Expo(X : Float) : Float; external 'dmath';
9101: { Exponential }
9102: function Exp2(X : Float) : Float; external 'dmath';
9103: { 2^X }
9104: function Exp10(X : Float) : Float; external 'dmath';
9105: { 10^X }
9106: function Log(X : Float) : Float; external 'dmath';
9107: { Natural log }
9108: function Log2(X : Float) : Float; external 'dmath';
9109: { Log, base 2 }
9110: function Log10(X : Float) : Float; external 'dmath';
9111: { Decimal log }
9112: function LogA(X, A : Float) : Float; external 'dmath';
9113: { Log, base A }
9114: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9115: { X^N }
9116: function Power(X, Y : Float) : Float; external 'dmath';
9117: { X^Y, X >= 0 }
9118: { ---------------------------------------------------------------
9119:   Trigonometric functions
9120:   --------------------------------------------------------------- }
9121: function Pythag(X, Y : Float) : Float; external 'dmath';
9122: { Sqrt(X^2 + Y^2) }
9123: function FixAngle(Theta : Float) : Float; external 'dmath';
9124: { Set Theta in -Pi..Pi }
9125: function Tan(X : Float) : Float; external 'dmath';
9126: { Tangent }
9127: function ArcSin(X : Float) : Float; external 'dmath';
9128: { Arc sinus }
9129: function ArcCos(X : Float) : Float; external 'dmath';
9130: { Arc cosinus }
9131: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9132: { Angle (Ox, OM) with M(X,Y) }
9133: { ---------------------------------------------------------------
9134:   Hyperbolic functions
9135:   --------------------------------------------------------------- }
9136: function Sinh(X : Float) : Float; external 'dmath';
9137: { Hyperbolic sine }
9138: function Cosh(X : Float) : Float; external 'dmath';
9139: { Hyperbolic cosine }
9140: function Tanh(X : Float) : Float; external 'dmath';
9141: { Hyperbolic tangent }
9142: function ArcSinh(X : Float) : Float; external 'dmath';
9143: { Inverse hyperbolic sine }
9144: function ArcCosh(X : Float) : Float; external 'dmath';
9145: { Inverse hyperbolic cosine }
9146: function ArcTanh(X : Float) : Float; external 'dmath';
9147: { Inverse hyperbolic tangent }
9148: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9149: { Sinh & Cosh }
9150: { ---------------------------------------------------------------
9151:   Gamma function and related functions
9152:   --------------------------------------------------------------- }
9153: function Fact(N : Integer) : Float; external 'dmath';
9154: { Factorial }
9155: function SgnGamma(X : Float) : Integer; external 'dmath';
9156: { Sign of Gamma function }
9157: function Gamma(X : Float) : Float; external 'dmath';
9158: { Gamma function }
9159: function LnGamma(X : Float) : Float; external 'dmath';
9160: { Logarithm of Gamma function }
9161: function Stirling(X : Float) : Float; external 'dmath';
9162: { Stirling's formula for the Gamma function }
```

```
9163: function StirLog(X : Float) : Float; external 'dmath';
9164: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9165: function DiGamma(X : Float ) : Float; external 'dmath';
9166: { Digamma function }
9167: function TriGamma(X : Float ) : Float; external 'dmath';
9168: { Trigamma function }
9169: function IGamma(A, X : Float) : Float; external 'dmath';
9170: { Incomplete Gamma function}
9171: function JGamma(A, X : Float) : Float; external 'dmath';
9172: { Complement of incomplete Gamma function }
9173: function InvGamma(A, P : Float) : Float; external 'dmath';
9174: { Inverse of incomplete Gamma function }
9175: function Erf(X : Float) : Float; external 'dmath';
9176: { Error function }
9177: function Erfc(X : Float) : Float; external 'dmath';
9178: { Complement of error function }
9179: { ------------------------------------------------------------
9180:   Beta function and related functions
9181:   ------------------------------------------------------------ }
9182: function Beta(X, Y : Float) : Float; external 'dmath';
9183: { Beta function }
9184: function IBeta(A, B, X : Float) : Float; external 'dmath';
9185: { Incomplete Beta function }
9186: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9187: { Inverse of incomplete Beta function }
9188: { ------------------------------------------------------------
9189:   Lambert's function
9190:   ------------------------------------------------------------ }
9191: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9192:  ------------------------------------------------------------
9193:   Binomial distribution
9194:   ------------------------------------------------------------ }
9195: function Binomial(N, K : Integer) : Float; external 'dmath';
9196: { Binomial coefficient C(N,K) }
9197: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9198: { Probability of binomial distribution }
9199: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9200: { Cumulative probability for binomial distrib. }
9201: { ------------------------------------------------------------
9202:   Poisson distribution
9203:   ------------------------------------------------------------ }
9204: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9205: { Probability of Poisson distribution }
9206: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9207: { Cumulative probability for Poisson distrib. }
9208: { ------------------------------------------------------------
9209:   Exponential distribution
9210:   ------------------------------------------------------------ }
9211: function DExpo(A, X : Float) : Float; external 'dmath';
9212: { Density of exponential distribution with parameter A }
9213: function FExpo(A, X : Float) : Float; external 'dmath';
9214: { Cumulative probability function for exponential dist. with parameter A }
9215: { ------------------------------------------------------------
9216:   Standard normal distribution
9217:   ------------------------------------------------------------ }
9218: function DNorm(X : Float) : Float; external 'dmath';
9219: { Density of standard normal distribution }
9220: function FNorm(X : Float) : Float; external 'dmath';
9221: { Cumulative probability for standard normal distrib. }
9222: function PNorm(X : Float) : Float; external 'dmath';
9223: { Prob(|U| > X) for standard normal distrib. }
9224: function InvNorm(P : Float) : Float; external 'dmath';
9225: { Inverse of standard normal distribution }
9226: { ------------------------------------------------------------
9227:   Student's distribution
9228:   ------------------------------------------------------------ }
9229: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9230: { Density of Student distribution with Nu d.o.f. }
9231: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9232: { Cumulative probability for Student distrib. with Nu d.o.f. }
9233: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9234: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9235: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9236: { Inverse of Student's t-distribution function }
9237: { ------------------------------------------------------------
9238:   Khi-2 distribution
9239:   ------------------------------------------------------------ }
9240: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9241: { Density of Khi-2 distribution with Nu d.o.f. }
9242: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9243: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9244: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9245: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9246: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9247: { Inverse of Khi-2 distribution function }
9248: { ------------------------------------------------------------
9249:   Fisher-Snedecor distribution
9250:   ------------------------------------------------------------ }
9251: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
```

```
9252: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9253: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9254: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9255: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9256: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9257: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9258: { Inverse of Snedecor's F-distribution function }
9259: { ------------------------------------------------------------
9260:   Beta distribution
9261:   ------------------------------------------------------------ }
9262: function DBeta(A, B, X : Float) : Float; external 'dmath';
9263: { Density of Beta distribution with parameters A and B }
9264: function FBeta(A, B, X : Float) : Float; external 'dmath';
9265: { Cumulative probability for Beta distrib. with param. A and B }
9266: { ------------------------------------------------------------
9267:   Gamma distribution
9268:   ------------------------------------------------------------ }
9269: function DGamma(A, B, X : Float) : Float; external 'dmath';
9270: { Density of Gamma distribution with parameters A and B }
9271: function FGamma(A, B, X : Float) : Float; external 'dmath';
9272: { Cumulative probability for Gamma distrib. with param. A and B }
9273: { ------------------------------------------------------------
9274:   Expression evaluation
9275:   ------------------------------------------------------------ }
9276: function InitEval : Integer; external 'dmath';
9277: { Initializes built-in functions and returns their number }
9278: function Eval(ExpressionString : String) : Float; external 'dmath';
9279: { Evaluates an expression at run-time }
9280: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9281: { Assigns a value to a variable }
9282: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9283: { Adds a function to the parser }
9284: { ------------------------------------------------------------
9285:   Matrices and linear equations
9286:   ------------------------------------------------------------ }
9287: procedure GaussJordan(A            : TMatrix;
9288:                       Lb, Ub1, Ub2 : Integer;
9289:                       var Det      : Float); external 'dmath';
9290: { Transforms a matrix according to the Gauss-Jordan method }
9291: procedure LinEq(A       : TMatrix;
9292:                 B       : TVector;
9293:                 Lb, Ub  : Integer;
9294:                 var Det : Float); external 'dmath';
9295: { Solves a linear system according to the Gauss-Jordan method }
9296: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9297: { Cholesky factorization of a positive definite symmetric matrix }
9298: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9299: { LU decomposition }
9300: procedure LU_Solve(A      : TMatrix;
9301:                    B      : TVector;
9302:                    Lb, Ub : Integer;
9303:                    X      : TVector); external 'dmath';
9304: { Solution of linear system from LU decomposition }
9305: procedure QR_Decomp(A           : TMatrix;
9306:                     Lb, Ub1, Ub2 : Integer;
9307:                     R           : TMatrix); external 'dmath';
9308: { QR decomposition }
9309: procedure QR_Solve(Q, R         : TMatrix;
9310:                    B           : TVector;
9311:                    Lb, Ub1, Ub2 : Integer;
9312:                    X           : TVector); external 'dmath';
9313: { Solution of linear system from QR decomposition }
9314: procedure SV_Decomp(A           : TMatrix;
9315:                     Lb, Ub1, Ub2 : Integer;
9316:                     S           : TVector;
9317:                     V           : TMatrix); external 'dmath';
9318: { Singular value decomposition }
9319: procedure SV_SetZero(S      : TVector;
9320:                      Lb, Ub : Integer;
9321:                      Tol    : Float); external 'dmath';
9322: { Set lowest singular values to zero }
9323: procedure SV_Solve(U           : TMatrix;
9324:                    S           : TVector;
9325:                    V           : TMatrix;
9326:                    B           : TVector;
9327:                    Lb, Ub1, Ub2 : Integer;
9328:                    X           : TVector); external 'dmath';
9329: { Solution of linear system from SVD }
9330: procedure SV_Approx(U           : TMatrix;
9331:                     S           : TVector;
9332:                     V           : TMatrix;
9333:                     Lb, Ub1, Ub2 : Integer;
9334:                     A           : TMatrix); external 'dmath';
9335: { Matrix approximation from SVD }
9336: procedure EigenVals(A      : TMatrix;
9337:                     Lb, Ub : Integer;
9338:                     Lambda : TCompVector); external 'dmath';
9339: { Eigenvalues of a general square matrix }
9340: procedure EigenVect(A      : TMatrix;
```

```
9341:                    Lb, Ub : Integer;
9342:                    Lambda : TCompVector;
9343:                    V      : TMatrix); external 'dmath';
9344: { Eigenvalues and eigenvectors of a general square matrix }
9345: procedure EigenSym(A      : TMatrix;
9346:                    Lb, Ub : Integer;
9347:                    Lambda : TVector;
9348:                    V      : TMatrix); external 'dmath';
9349: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9350: procedure Jacobi(A              : TMatrix;
9351:                  Lb, Ub, MaxIter : Integer;
9352:                  Tol             : Float;
9353:                  Lambda          : TVector;
9354:                  V               : TMatrix); external 'dmath';
9355: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9356: { ---------------------------------------------------------------
9357:   Optimization
9358:   --------------------------------------------------------------- }
9359: procedure MinBrack(Func                    : TFunc;
9360:                    var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9361: { Brackets a minimum of a function }
9362: procedure GoldSearch(Func           : TFunc;
9363:                      A, B           : Float;
9364:                      MaxIter        : Integer;
9365:                      Tol            : Float;
9366:                      var Xmin, Ymin : Float); external 'dmath';
9367: { Minimization of a function of one variable (golden search) }
9368: procedure LinMin(Func      : TFuncNVar;
9369:                  X, DeltaX : TVector;
9370:                  Lb, Ub    : Integer;
9371:                  var R     : Float;
9372:                  MaxIter   : Integer;
9373:                  Tol       : Float;
9374:                  var F_min : Float); external 'dmath';
9375: { Minimization of a function of several variables along a line }
9376: procedure Newton(Func      : TFuncNVar;
9377:                  HessGrad  : THessGrad;
9378:                  X         : TVector;
9379:                  Lb, Ub    : Integer;
9380:                  MaxIter   : Integer;
9381:                  Tol       : Float;
9382:                  var F_min : Float;
9383:                  G         : TVector;
9384:                  H_inv     : TMatrix;
9385:                  var Det   : Float); external 'dmath';
9386: { Minimization of a function of several variables (Newton's method) }
9387: procedure SaveNewton(FileName : string); external 'dmath';
9388: { Save Newton iterations in a file }
9389: procedure Marquardt(Func      : TFuncNVar;
9390:                     HessGrad  : THessGrad;
9391:                     X         : TVector;
9392:                     Lb, Ub    : Integer;
9393:                     MaxIter   : Integer;
9394:                     Tol       : Float;
9395:                     var F_min : Float;
9396:                     G         : TVector;
9397:                     H_inv     : TMatrix;
9398:                     var Det   : Float); external 'dmath';
9399: { Minimization of a function of several variables (Marquardt's method) }
9400: procedure SaveMarquardt(FileName : string); external 'dmath';
9401: { Save Marquardt iterations in a file }
9402: procedure BFGS(Func      : TFuncNVar;
9403:                Gradient  : TGradient;
9404:                X         : TVector;
9405:                Lb, Ub    : Integer;
9406:                MaxIter   : Integer;
9407:                Tol       : Float;
9408:                var F_min : Float;
9409:                G         : TVector;
9410:                H_inv     : TMatrix); external 'dmath';
9411: { Minimization of a function of several variables (BFGS method) }
9412: procedure SaveBFGS(FileName : string); external 'dmath';
9413: { Save BFGS iterations in a file }
9414: procedure Simplex(Func      : TFuncNVar;
9415:                   X         : TVector;
9416:                   Lb, Ub    : Integer;
9417:                   MaxIter   : Integer;
9418:                   Tol       : Float;
9419:                   var F_min : Float); external 'dmath';
9420: { Minimization of a function of several variables (Simplex) }
9421: procedure SaveSimplex(FileName : string); external 'dmath';
9422: { Save Simplex iterations in a file }
9423: { ---------------------------------------------------------------
9424:   Nonlinear equations
9425:   --------------------------------------------------------------- }
9426: procedure RootBrack(Func              : TFunc;
9427:                     var X, Y, FX, FY : Float); external 'dmath';
9428: { Brackets a root of function Func between X and Y }
9429: procedure Bisect(Func     : TFunc;
```

```
9430:                 var X, Y : Float;
9431:                 MaxIter  : Integer;
9432:                 Tol      : Float;
9433:                 var F    : Float); external 'dmath';
9434: { Bisection method }
9435: procedure Secant(Func     : TFunc;
9436:                 var X, Y : Float;
9437:                 MaxIter  : Integer;
9438:                 Tol      : Float;
9439:                 var F    : Float); external 'dmath';
9440: { Secant method }
9441: procedure NewtEq(Func, Deriv : TFunc;
9442:                 var X       : Float;
9443:                 MaxIter     : Integer;
9444:                 Tol         : Float;
9445:                 var F       : Float); external 'dmath';
9446: { Newton-Raphson method for a single nonlinear equation }
9447: procedure NewtEqs(Equations : TEquations;
9448:                   Jacobian  : TJacobian;
9449:                   X, F      : TVector;
9450:                   Lb, Ub    : Integer;
9451:                   MaxIter   : Integer;
9452:                   Tol       : Float); external 'dmath';
9453: { Newton-Raphson method for a system of nonlinear equations }
9454: procedure Broyden(Equations : TEquations;
9455:                   X, F      : TVector;
9456:                   Lb, Ub    : Integer;
9457:                   MaxIter   : Integer;
9458:                   Tol       : Float); external 'dmath';
9459: { Broyden's method for a system of nonlinear equations }
9460: { ------------------------------------------------------------------
9461:   Polynomials and rational fractions
9462:   ------------------------------------------------------------------ }
9463: function Poly(X    : Float;
9464:               Coef : TVector;
9465:               Deg  : Integer) : Float; external 'dmath';
9466: { Evaluates a polynomial }
9467: function RFrac(X          : Float;
9468:                Coef       : TVector;
9469:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9470: { Evaluates a rational fraction }
9471: function RootPol1(A, B  : Float;
9472:                   var X : Float) : Integer; external 'dmath';
9473: { Solves the linear equation A + B * X = 0 }
9474: function RootPol2(Coef : TVector;
9475:                   Z    : TCompVector) : Integer; external 'dmath';
9476: { Solves a quadratic equation }
9477: function RootPol3(Coef : TVector;
9478:                   Z    : TCompVector) : Integer; external 'dmath';
9479: { Solves a cubic equation }
9480: function RootPol4(Coef : TVector;
9481:                   Z    : TCompVector) : Integer; external 'dmath';
9482: { Solves a quartic equation }
9483: function RootPol(Coef : TVector;
9484:                  Deg  : Integer;
9485:                  Z    : TCompVector) : Integer; external 'dmath';
9486: { Solves a polynomial equation }
9487: function SetRealRoots(Deg : Integer;
9488:                       Z   : TCompVector;
9489:                       Tol : Float) : Integer; external 'dmath';
9490: { Set the imaginary part of a root to zero }
9491: procedure SortRoots(Deg : Integer;
9492:                     Z   : TCompVector); external 'dmath';
9493: { Sorts the roots of a polynomial }
9494: { ------------------------------------------------------------------
9495:   Numerical integration and differential equations
9496:   ------------------------------------------------------------------ }
9497: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9498: { Integration by trapezoidal rule }
9499: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9500: { Integral from A to B }
9501: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9502: { Integral from 0 to B }
9503: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9504: { Convolution product at time T }
9505: procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9506: { Convolution by trapezoidal rule }
9507: procedure RKF45(F                    : TDiffEqs;
9508:                 Neqn                 : Integer;
9509:                 Y, Yp                : TVector;
9510:                 var T                : Float;
9511:                 Tout, RelErr, AbsErr : Float;
9512:                 var Flag             : Integer); external 'dmath';
9513: { Integration of a system of differential equations }
9514: { ------------------------------------------------------------------
9515:   Fast Fourier Transform
9516:   ------------------------------------------------------------------ }
9517: procedure FFT(NumSamples        : Integer;
9518:               InArray, OutArray : TCompVector); external 'dmath';
```

```
9519: { Fast Fourier Transform }
9520: procedure IFFT(NumSamples        : Integer;
9521:                InArray, OutArray : TCompVector); external 'dmath';
9522: { Inverse Fast Fourier Transform }
9523: procedure FFT_Integer(NumSamples     : Integer;
9524:                       RealIn, ImagIn : TIntVector;
9525:                       OutArray       : TCompVector); external 'dmath';
9526: { Fast Fourier Transform for integer data }
9527: procedure FFT_Integer_Cleanup; external 'dmath';
9528: { Clear memory after a call to FFT_Integer }
9529: procedure CalcFrequency(NumSamples,
9530:                         FrequencyIndex : Integer;
9531:                         InArray        : TCompVector;
9532:                         var FFT        : Complex); external 'dmath';
9533: { Direct computation of Fourier transform }
9534: { ---------------------------------------------------------------
9535:   Random numbers
9536:   --------------------------------------------------------------- }
9537: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9538: { Select generator }
9539: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9540: { Initialize generator }
9541: function IRanGen : RNG_IntType; external 'dmath';
9542: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9543: function IRanGen31 : RNG_IntType; external 'dmath';
9544: { 31-bit random integer in [0 .. 2^31 - 1] }
9545: function RanGen1 : Float; external 'dmath';
9546: { 32-bit random real in [0,1] }
9547: function RanGen2 : Float; external 'dmath';
9548: { 32-bit random real in [0,1) }
9549: function RanGen3 : Float; external 'dmath';
9550: { 32-bit random real in (0,1) }
9551: function RanGen53 : Float; external 'dmath';
9552: { 53-bit random real in [0,1) }
9553: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9554: { Initializes the 'Multiply with carry' random number generator }
9555: function IRanMWC : RNG_IntType; external 'dmath';
9556: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9557: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9558: { Initializes Mersenne Twister generator with a seed }
9559: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9560:                         KeyLength : Word); external 'dmath';
9561: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9562: function IRanMT : RNG_IntType; external 'dmath';
9563: { Random integer from MT generator }
9564: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9565: { Initializes the UVAG generator with a string }
9566: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9567: { Initializes the UVAG generator with an integer }
9568: function IRanUVAG : RNG_IntType; external 'dmath';
9569: { Random integer from UVAG generator }
9570: function RanGaussStd : Float; external 'dmath';
9571: { Random number from standard normal distribution }
9572: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9573: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9574: procedure RanMult(M      : TVector; L      : TMatrix;
9575:                   Lb, Ub : Integer;
9576:                   X      : TVector); external 'dmath';
9577: { Random vector from multinormal distribution (correlated) }
9578: procedure RanMultIndep(M, S   : TVector;
9579:                        Lb, Ub : Integer;
9580:                        X      : TVector); external 'dmath';
9581: { Random vector from multinormal distribution (uncorrelated) }
9582: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9583: { Initializes Metropolis-Hastings parameters }
9584: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9585: { Returns Metropolis-Hastings parameters }
9586: procedure Hastings(Func      : TFuncNVar;
9587:                    T         : Float;
9588:                    X         : TVector;
9589:                    V         : TMatrix;
9590:                    Lb, Ub    : Integer;
9591:                    Xmat      : TMatrix;
9592:                    X_min     : TVector;
9593:                    var F_min : Float); external 'dmath';
9594: { Simulation of a probability density function by Metropolis-Hastings }
9595: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9596: { Initializes Simulated Annealing parameters }
9597: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9598: { Initializes log file }
9599: procedure SimAnn(Func          : TFuncNVar;
9600:                  X, Xmin, Xmax : TVector;
9601:                  Lb, Ub        : Integer;
9602:                  var F_min     : Float); external 'dmath';
9603: { Minimization of a function of several var. by simulated annealing }
9604: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9605: { Initializes Genetic Algorithm parameters }
9606: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9607: { Initializes log file }
```

```
9608: procedure GenAlg(Func          : TFuncNVar;
9609:                  X, Xmin, Xmax : TVector;
9610:                  Lb, Ub        : Integer;
9611:                  var F_min     : Float); external 'dmath';
9612: { Minimization of a function of several var. by genetic algorithm }
9613: { ---------------------------------------------------------------
9614:   Statistics
9615:   --------------------------------------------------------------- }
9616: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9617: { Mean of sample X }
9618: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9619: { Minimum of sample X }
9620: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9621: { Maximum of sample X }
9622: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9623: { Median of sample X }
9624: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9625: { Standard deviation estimated from sample X }
9626: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9627: { Standard deviation of population }
9628: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9629: { Correlation coefficient }
9630: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9631: { Skewness of sample X }
9632: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9633: { Kurtosis of sample X }
9634: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9635: { Quick sort (ascending order) }
9636: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9637: { Quick sort (descending order) }
9638: procedure Interval(X1, X2             : Float;
9639:                    MinDiv, MaxDiv     : Integer;
9640:                    var Min, Max, Step : Float); external 'dmath';
9641: { Determines an interval for a set of values }
9642: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9643:                     var XMin, XMax, XStep : Float); external 'dmath';
9644: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9645: procedure StudIndep(N1, N2         : Integer;
9646:                     M1, M2, S1, S2 : Float;
9647:                     var T          : Float;
9648:                     var DoF        : Integer); external 'dmath';
9649: { Student t-test for independent samples }
9650: procedure StudPaired(X, Y    : TVector;
9651:                      Lb, Ub  : Integer;
9652:                      var T   : Float;
9653:                      var DoF : Integer); external 'dmath';
9654: { Student t-test for paired samples }
9655: procedure AnOVa1(Ns               : Integer;
9656:                  N                : TIntVector;
9657:                  M, S             : TVector;
9658:                  var V_f, V_r, F  : Float;
9659:                  var DoF_f, DoF_r : Integer); external 'dmath';
9660: { One-way analysis of variance }
9661: procedure AnOVa2(NA, NB, Nobs : Integer;
9662:                  M, S         : TMatrix;
9663:                  V, F         : TVector;
9664:                  DoF          : TIntVector); external 'dmath';
9665: { Two-way analysis of variance }
9666: procedure Snedecor(N1, N2         : Integer;
9667:                    S1, S2         : Float;
9668:                    var F          : Float;
9669:                    var DoF1, DoF2 : Integer); external 'dmath';
9670: { Snedecor's F-test (comparison of two variances) }
9671: procedure Bartlett(Ns       : Integer;
9672:                    N        : TIntVector;
9673:                    S        : TVector;
9674:                    var Khi2 : Float;
9675:                    var DoF  : Integer); external 'dmath';
9676: { Bartlett's test (comparison of several variances) }
9677: procedure Mann_Whitney(N1, N2     : Integer;
9678:                        X1, X2     : TVector;
9679:                        var U, Eps : Float); external 'dmath';
9680: { Mann-Whitney test}
9681: procedure Wilcoxon(X, Y       : TVector;
9682:                    Lb, Ub     : Integer;
9683:                    var Ndiff  : Integer;
9684:                    var T, Eps : Float); external 'dmath';
9685: { Wilcoxon test }
9686: procedure Kruskal_Wallis(Ns      : Integer;
9687:                          N       : TIntVector;
9688:                          X       : TMatrix;
9689:                          var H   : Float;
9690:                          var DoF : Integer); external 'dmath';
9691: { Kruskal-Wallis test }
9692: procedure Khi2_Conform(N_cls    : Integer;
9693:                        N_estim  : Integer;
9694:                        Obs      : TIntVector;
9695:                        Calc     : TVector;
9696:                        var Khi2 : Float;
```

```
9697:                       var DoF  : Integer); external 'dmath';
9698: { Khi-2 test for conformity }
9699: procedure Khi2_Indep(N_lin    : Integer;
9700:                      N_col    : Integer;
9701:                      Obs      : TIntMatrix;
9702:                      var Khi2 : Float;
9703:                      var DoF  : Integer); external 'dmath';
9704: { Khi-2 test for independence }
9705: procedure Woolf_Conform(N_cls   : Integer;
9706:                        N_estim : Integer;
9707:                        Obs     : TIntVector;
9708:                        Calc    : TVector;
9709:                        var G   : Float;
9710:                        var DoF : Integer); external 'dmath';
9711: { Woolf's test for conformity }
9712: procedure Woolf_Indep(N_lin   : Integer;
9713:                       N_col   : Integer;
9714:                       Obs     : TIntMatrix;
9715:                       var G   : Float;
9716:                       var DoF : Integer); external 'dmath';
9717: { Woolf's test for independence }
9718: procedure DimStatClassVector(var C : TStatClassVector;
9719:                              Ub    : Integer); external 'dmath';
9720: { Allocates an array of statistical classes: C[0..Ub] }
9721: procedure Distrib(X       : TVector;
9722:                   Lb, Ub  : Integer;
9723:                   A, B, H : Float;
9724:                   C       : TStatClassVector); external 'dmath';
9725: { Distributes an array X[Lb..Ub] into statistical classes }
9726: { ----------------------------------------------------------------
9727:   Linear / polynomial regression
9728:   ---------------------------------------------------------------- }
9729: procedure LinFit(X, Y   : TVector;
9730:                  Lb, Ub : Integer;
9731:                  B      : TVector;
9732:                  V      : TMatrix); external 'dmath';
9733: { Linear regression : Y = B(0) + B(1) * X }
9734: procedure WLinFit(X, Y, S : TVector;
9735:                   Lb, Ub  : Integer;
9736:                   B       : TVector;
9737:                   V       : TMatrix); external 'dmath';
9738: { Weighted linear regression : Y = B(0) + B(1) * X }
9739: procedure SVDLinFit(X, Y   : TVector;
9740:                     Lb, Ub : Integer;
9741:                     SVDTol : Float;
9742:                     B      : TVector;
9743:                     V      : TMatrix); external 'dmath';
9744: { Unweighted linear regression by singular value decomposition }
9745: procedure WSVDLinFit(X, Y, S : TVector;
9746:                      Lb, Ub  : Integer;
9747:                      SVDTol  : Float;
9748:                      B       : TVector;
9749:                      V       : TMatrix); external 'dmath';
9750: { Weighted linear regression by singular value decomposition }
9751: procedure MulFit(X            : TMatrix;
9752:                  Y            : TVector;
9753:                  Lb, Ub, Nvar : Integer;
9754:                  ConsTerm     : Boolean;
9755:                  B            : TVector;
9756:                  V            : TMatrix); external 'dmath';
9757: { Multiple linear regression by Gauss-Jordan method }
9758: procedure WMulFit(X            : TMatrix;
9759:                   Y, S         : TVector;
9760:                   Lb, Ub, Nvar : Integer;
9761:                   ConsTerm     : Boolean;
9762:                   B            : TVector;
9763:                   V            : TMatrix); external 'dmath';
9764: { Weighted multiple linear regression by Gauss-Jordan method }
9765: procedure SVDFit(X            : TMatrix;
9766:                  Y            : TVector;
9767:                  Lb, Ub, Nvar : Integer;
9768:                  ConsTerm     : Boolean;
9769:                  SVDTol       : Float;
9770:                  B            : TVector;
9771:                  V            : TMatrix); external 'dmath';
9772: { Multiple linear regression by singular value decomposition }
9773: procedure WSVDFit(X            : TMatrix;
9774:                   Y, S         : TVector;
9775:                   Lb, Ub, Nvar : Integer;
9776:                   ConsTerm     : Boolean;
9777:                   SVDTol       : Float;
9778:                   B            : TVector;
9779:                   V            : TMatrix); external 'dmath';
9780: { Weighted multiple linear regression by singular value decomposition }
9781: procedure PolFit(X, Y        : TVector;
9782:                  Lb, Ub, Deg : Integer;
9783:                  B           : TVector;
9784:                  V           : TMatrix); external 'dmath';
9785: { Polynomial regression by Gauss-Jordan method }
```

```
9786: procedure WPolFit(X, Y, S     : TVector;
9787:                   Lb, Ub, Deg : Integer;
9788:                   B           : TVector;
9789:                   V           : TMatrix); external 'dmath';
9790: { Weighted polynomial regression by Gauss-Jordan method }
9791: procedure SVDPolFit(X, Y        : TVector;
9792:                     Lb, Ub, Deg : Integer;
9793:                     SVDTol      : Float;
9794:                     B           : TVector;
9795:                     V           : TMatrix); external 'dmath';
9796: { Unweighted polynomial regression by singular value decomposition }
9797: procedure WSVDPolFit(X, Y, S     : TVector;
9798:                      Lb, Ub, Deg : Integer;
9799:                      SVDTol      : Float;
9800:                      B           : TVector;
9801:                      V           : TMatrix); external 'dmath';
9802: { Weighted polynomial regression by singular value decomposition }
9803: procedure RegTest(Y, Ycalc : TVector;
9804:                   LbY, UbY : Integer;
9805:                   V        : TMatrix;
9806:                   LbV, UbV : Integer;
9807:                   var Test : TRegTest); external 'dmath';
9808: { Test of unweighted regression }
9809: procedure WRegTest(Y, Ycalc, S : TVector;
9810:                    LbY, UbY    : Integer;
9811:                    V           : TMatrix;
9812:                    LbV, UbV    : Integer;
9813:                    var Test    : TRegTest); external 'dmath';
9814: { Test of weighted regression }
9815: { ---------------------------------------------------------------
9816:   Nonlinear regression
9817:   --------------------------------------------------------------- }
9818: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9819: { Sets the optimization algorithm for nonlinear regression }
9820: function GetOptAlgo : TOptAlgo; external 'dmath';
9821: { Returns the optimization algorithm }
9822: procedure SetMaxParam(N : Byte); external 'dmath';
9823: { Sets the maximum number of regression parameters for nonlinear regression }
9824: function GetMaxParam : Byte; external 'dmath';
9825: { Returns the maximum number of regression parameters for nonlinear regression }
9826: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9827: { Sets the bounds on the I-th regression parameter }
9828: procedure GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
9829: { Returns the bounds on the I-th regression parameter }
9830: procedure NLFit(RegFunc   : TRegFunc;
9831:                 DerivProc : TDerivProc;
9832:                 X, Y      : TVector;
9833:                 Lb, Ub    : Integer;
9834:                 MaxIter   : Integer;
9835:                 Tol       : Float;
9836:                 B         : TVector;
9837:                 FirstPar,
9838:                 LastPar   : Integer;
9839:                 V         : TMatrix); external 'dmath';
9840: { Unweighted nonlinear regression }
9841: procedure WNLFit(RegFunc   : TRegFunc;
9842:                  DerivProc : TDerivProc;
9843:                  X, Y, S   : TVector;
9844:                  Lb, Ub    : Integer;
9845:                  MaxIter   : Integer;
9846:                  Tol       : Float;
9847:                  B         : TVector;
9848:                  FirstPar,
9849:                  LastPar   : Integer;
9850:                  V         : TMatrix); external 'dmath';
9851: { Weighted nonlinear regression }
9852: procedure SetMCFile(FileName : String); external 'dmath';
9853: { Set file for saving MCMC simulations }
9854: procedure SimFit(RegFunc   : TRegFunc;
9855:                  X, Y      : TVector;
9856:                  Lb, Ub    : Integer;
9857:                  B         : TVector;
9858:                  FirstPar,
9859:                  LastPar   : Integer;
9860:                  V         : TMatrix); external 'dmath';
9861: { Simulation of unweighted nonlinear regression by MCMC }
9862: procedure WSimFit(RegFunc   : TRegFunc;
9863:                   X, Y, S   : TVector;
9864:                   Lb, Ub    : Integer;
9865:                   B         : TVector;
9866:                   FirstPar,
9867:                   LastPar   : Integer;
9868:                   V         : TMatrix); external 'dmath';
9869: { Simulation of weighted nonlinear regression by MCMC }
9870: { ---------------------------------------------------------------
9871:   Nonlinear regression models
9872:   --------------------------------------------------------------- }
9873: procedure FracFit(X, Y       : TVector;
9874:                   Lb, Ub     : Integer;
```

```
9875:                  Deg1, Deg2 : Integer;
9876:                  ConsTerm   : Boolean;
9877:                  MaxIter    : Integer;
9878:                  Tol        : Float;
9879:                  B          : TVector;
9880:                  V          : TMatrix); external 'dmath';
9881: { Unweighted fit of rational fraction }
9882: procedure WFracFit(X, Y, S    : TVector;
9883:                   Lb, Ub     : Integer;
9884:                   Deg1, Deg2 : Integer;
9885:                   ConsTerm   : Boolean;
9886:                   MaxIter    : Integer;
9887:                   Tol        : Float;
9888:                   B          : TVector;
9889:                   V          : TMatrix); external 'dmath';
9890: { Weighted fit of rational fraction }
9891:
9892: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9893: { Returns the value of the rational fraction at point X }
9894: procedure ExpFit(X, Y         : TVector;
9895:                  Lb, Ub, Nexp : Integer;
9896:                  ConsTerm     : Boolean;
9897:                  MaxIter      : Integer;
9898:                  Tol          : Float;
9899:                  B            : TVector;
9900:                  V            : TMatrix); external 'dmath';
9901: { Unweighted fit of sum of exponentials }
9902: procedure WExpFit(X, Y, S      : TVector;
9903:                   Lb, Ub, Nexp : Integer;
9904:                   ConsTerm     : Boolean;
9905:                   MaxIter      : Integer;
9906:                   Tol          : Float;
9907:                   B            : TVector;
9908:                   V            : TMatrix); external 'dmath';
9909: { Weighted fit of sum of exponentials }
9910: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9911: { Returns the value of the regression function at point X }
9912: procedure IncExpFit(X, Y     : TVector;
9913:                     Lb, Ub   : Integer;
9914:                     ConsTerm : Boolean;
9915:                     MaxIter  : Integer;
9916:                     Tol      : Float;
9917:                     B        : TVector;
9918:                     V        : TMatrix); external 'dmath';
9919: { Unweighted fit of model of increasing exponential }
9920: procedure WIncExpFit(X, Y, S  : TVector;
9921:                      Lb, Ub   : Integer;
9922:                      ConsTerm : Boolean;
9923:                      MaxIter  : Integer;
9924:                      Tol      : Float;
9925:                      B        : TVector;
9926:                      V        : TMatrix); external 'dmath';
9927: { Weighted fit of increasing exponential }
9928: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9929: { Returns the value of the regression function at point X }
9930: procedure ExpLinFit(X, Y    : TVector;
9931:                     Lb, Ub  : Integer;
9932:                     MaxIter : Integer;
9933:                     Tol     : Float;
9934:                     B       : TVector;
9935:                     V       : TMatrix); external 'dmath';
9936: { Unweighted fit of the "exponential + linear" model }
9937: procedure WExpLinFit(X, Y, S : TVector;
9938:                      Lb, Ub  : Integer;
9939:                      MaxIter : Integer;
9940:                      Tol     : Float;
9941:                      B       : TVector;
9942:                      V       : TMatrix); external 'dmath';
9943: { Weighted fit of the "exponential + linear" model }
9944:
9945: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9946: { Returns the value of the regression function at point X }
9947: procedure MichFit(X, Y    : TVector;
9948:                   Lb, Ub  : Integer;
9949:                   MaxIter : Integer;
9950:                   Tol     : Float;
9951:                   B       : TVector;
9952:                   V       : TMatrix); external 'dmath';
9953: { Unweighted fit of Michaelis equation }
9954: procedure WMichFit(X, Y, S : TVector;
9955:                    Lb, Ub  : Integer;
9956:                    MaxIter : Integer;
9957:                    Tol     : Float;
9958:                    B       : TVector;
9959:                    V       : TMatrix); external 'dmath';
9960: { Weighted fit of Michaelis equation }
9961: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9962: { Returns the value of the Michaelis equation at point X }
9963: procedure MintFit(X, Y    : TVector;
```

```
 9964:                   Lb, Ub  : Integer;
 9965:                   MintVar : TMintVar;
 9966:                   Fit_S0  : Boolean;
 9967:                   MaxIter : Integer;
 9968:                   Tol     : Float;
 9969:                   B       : TVector;
 9970:                   V       : TMatrix); external 'dmath';
 9971: { Unweighted fit of the integrated Michaelis equation }
 9972: procedure WMintFit(X, Y, S : TVector;
 9973:                    Lb, Ub  : Integer;
 9974:                    MintVar : TMintVar;
 9975:                    Fit_S0  : Boolean;
 9976:                    MaxIter : Integer;
 9977:                    Tol     : Float;
 9978:                    B       : TVector;
 9979:                    V       : TMatrix); external 'dmath';
 9980: { Weighted fit of the integrated Michaelis equation }
 9981: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9982: { Returns the value of the integrated Michaelis equation at point X }
 9983: procedure HillFit(X, Y    : TVector;
 9984:                   Lb, Ub  : Integer;
 9985:                   MaxIter : Integer;
 9986:                   Tol     : Float;
 9987:                   B       : TVector;
 9988:                   V       : TMatrix); external 'dmath';
 9989: { Unweighted fit of Hill equation }
 9990: procedure WHillFit(X, Y, S : TVector;
 9991:                    Lb, Ub  : Integer;
 9992:                    MaxIter : Integer;
 9993:                    Tol     : Float;
 9994:                    B       : TVector;
 9995:                    V       : TMatrix); external 'dmath';
 9996: { Weighted fit of Hill equation }
 9997: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9998: { Returns the value of the Hill equation at point X }
 9999: procedure LogiFit(X, Y     : TVector;
10000:                   Lb, Ub   : Integer;
10001:                   ConsTerm : Boolean;
10002:                   General  : Boolean;
10003:                   MaxIter  : Integer;
10004:                   Tol      : Float;
10005:                   B        : TVector;
10006:                   V        : TMatrix); external 'dmath';
10007: { Unweighted fit of logistic function }
10008: procedure WLogiFit(X, Y, S  : TVector;
10009:                    Lb, Ub   : Integer;
10010:                    ConsTerm : Boolean;
10011:                    General  : Boolean;
10012:                    MaxIter  : Integer;
10013:                    Tol      : Float;
10014:                    B        : TVector;
10015:                    V        : TMatrix); external 'dmath';
10016: { Weighted fit of logistic function }
10017: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10018: { Returns the value of the logistic function at point X }
10019: procedure PKFit(X, Y    : TVector;
10020:                 Lb, Ub  : Integer;
10021:                 MaxIter : Integer;
10022:                 Tol     : Float;
10023:                 B       : TVector;
10024:                 V       : TMatrix); external 'dmath';
10025: { Unweighted fit of the acid-base titration curve }
10026: procedure WPKFit(X, Y, S : TVector;
10027:                  Lb, Ub  : Integer;
10028:                  MaxIter : Integer;
10029:                  Tol     : Float;
10030:                  B       : TVector;
10031:                  V       : TMatrix); external 'dmath';
10032: { Weighted fit of the acid-base titration curve }
10033: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10034: { Returns the value of the acid-base titration function at point X }
10035: procedure PowFit(X, Y    : TVector;
10036:                  Lb, Ub  : Integer;
10037:                  MaxIter : Integer;
10038:                  Tol     : Float;
10039:                  B       : TVector;
10040:                  V       : TMatrix); external 'dmath';
10041: { Unweighted fit of power function }
10042: procedure WPowFit(X, Y, S : TVector;
10043:                   Lb, Ub  : Integer;
10044:                   MaxIter : Integer;
10045:                   Tol     : Float;
10046:                   B       : TVector;
10047:                   V       : TMatrix); external 'dmath';
10048: { Weighted fit of power function }
10049:
10050: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10051: { Returns the value of the power function at point X }
10052: procedure GammaFit(X, Y    : TVector;
```

```
10053:                   Lb, Ub  : Integer;
10054:                   MaxIter : Integer;
10055:                   Tol     : Float;
10056:                   B       : TVector;
10057:                   V       : TMatrix); external 'dmath';
10058: { Unweighted fit of gamma distribution function }
10059: procedure WGammaFit(X, Y, S : TVector;
10060:                    Lb, Ub  : Integer;
10061:                    MaxIter : Integer;
10062:                    Tol     : Float;
10063:                    B       : TVector;
10064:                    V       : TMatrix); external 'dmath';
10065: { Weighted fit of gamma distribution function }
10066: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10067: { Returns the value of the gamma distribution function at point X }
10068: { ---------------------------------------------------------------
10069:   Principal component analysis
10070:   --------------------------------------------------------------- }
10071: procedure VecMean(X            : TMatrix;
10072:                  Lb, Ub, Nvar : Integer;
10073:                  M            : TVector); external 'dmath';
10074: { Computes the mean vector M from matrix X }
10075: procedure VecSD(X            : TMatrix;
10076:                Lb, Ub, Nvar : Integer;
10077:                M, S         : TVector); external 'dmath';
10078: { Computes the vector of standard deviations S from matrix X }
10079: procedure MatVarCov(X            : TMatrix;
10080:                    Lb, Ub, Nvar : Integer;
10081:                    M            : TVector;
10082:                    V            : TMatrix); external 'dmath';
10083: { Computes the variance-covariance matrix V from matrix X }
10084: procedure MatCorrel(V    : TMatrix;
10085:                    Nvar : Integer;
10086:                    R    : TMatrix); external 'dmath';
10087: { Computes the correlation matrix R from the var-cov matrix V }
10088: procedure PCA(R      : TMatrix;
10089:               Nvar   : Integer;
10090:               Lambda : TVector;
10091:               C, Rc  : TMatrix); external 'dmath';
10092: { Performs a principal component analysis of the correlation matrix R }
10093: procedure ScaleVar(X            : TMatrix;
10094:                    Lb, Ub, Nvar : Integer;
10095:                    M, S         : TVector;
10096:                    Z            : TMatrix); external 'dmath';
10097: { Scales a set of variables by subtracting means and dividing by SD's }
10098: procedure PrinFac(Z            : TMatrix;
10099:                  Lb, Ub, Nvar : Integer;
10100:                  C, F         : TMatrix); external 'dmath';
10101: { Computes principal factors }
10102: { ---------------------------------------------------------------
10103:   Strings
10104:   --------------------------------------------------------------- }
10105: function LTrim(S : String) : String; external 'dmath';
10106: { Removes leading blanks }
10107: function RTrim(S : String) : String; external 'dmath';
10108: { Removes trailing blanks }
10109: function Trim(S : String) : String; external 'dmath';
10110: { Removes leading and trailing blanks }
10111: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10112: { Returns a string made of character C repeated N times }
10113: function RFill(S : String; L : Byte) : String; external 'dmath';
10114: { Completes string S with trailing blanks for a total length L }
10115: function LFill(S : String; L : Byte) : String; external 'dmath';
10116: { Completes string S with leading blanks for a total length L }
10117: function CFill(S : String; L : Byte) : String; external 'dmath';
10118: { Centers string S on a total length L }
10119: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10120: { Replaces in string S all the occurences of C1 by C2 }
10121: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10122: { Extracts a field from a string }
10123: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10124: { Parses a string into its constitutive fields }
10125: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10126: { Sets the numeric format }
10127: function FloatStr(X : Float) : String; external 'dmath';
10128: { Converts a real to a string according to the numeric format }
10129: function IntStr(N : LongInt) : String; external 'dmath';
10130: { Converts an integer to a string }
10131: function CompStr(Z : Complex) : String; external 'dmath';
10132: { Converts a complex number to a string }
10133: {$IFDEF DELPHI}
10134: function StrDec(S : String) : String; external 'dmath';
10135: { Set decimal separator to the symbol defined in SysUtils }
10136: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10137: { Test if a string represents a number and returns it in X }
10138: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10139: { Reads a floating point number from an Edit control }
10140: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10141: { Writes a floating point number in a text file }
```

```
10142: {$ENDIF}
10143: { ---------------------------------------------------------------
10144:   BGI / Delphi graphics
10145:   --------------------------------------------------------------- }
10146: function InitGraphics
10147: {$IFDEF DELPHI}
10148: (Width, Height : Integer) : Boolean;
10149: {$ELSE}
10150: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10151: { Enters graphic mode }
10152: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10153:                     X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10154: { Sets the graphic window }
10155: procedure SetOxScale(Scale                : TScale;
10156:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10157: { Sets the scale on the Ox axis }
10158: procedure SetOyScale(Scale                : TScale;
10159:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10160: { Sets the scale on the Oy axis }
10161: procedure GetOxScale(var Scale                : TScale;
10162:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10163: { Returns the scale on the Ox axis }
10164: procedure GetOyScale(var Scale                : TScale;
10165:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10166: { Returns the scale on the Oy axis }
10167: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10168: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10169: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10170: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10171: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10172: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10173: {$IFNDEF DELPHI}
10174: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10175: { Sets the font for the main graph title }
10176: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10177: { Sets the font for the Ox axis (title and labels) }
10178: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10179: { Sets the font for the Oy axis (title and labels) }
10180: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10181: { Sets the font for the legends }
10182: procedure SetClipping(Clip : Boolean); external 'dmath';
10183: { Determines whether drawings are clipped at the current viewport
10184:   boundaries, according to the value of the Boolean parameter Clip }
10185: {$ENDIF}
10186: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10187: { Plots the horizontal axis }
10188: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10189: { Plots the vertical axis }
10190: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10191: { Plots a grid on the graph }
10192: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10193: { Writes the title of the graph }
10194: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10195: { Sets the maximum number of curves and re-initializes their parameters }
10196: procedure SetPointParam
10197: {$IFDEF DELPHI}
10198: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10199: {$ELSE}
10200: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10201: { Sets the point parameters for curve # CurvIndex }
10202: procedure SetLineParam
10203: {$IFDEF DELPHI}
10204: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10205: {$ELSE}
10206: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10207: { Sets the line parameters for curve # CurvIndex }
10208: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10209: { Sets the legend for curve # CurvIndex }
10210: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10211: { Sets the step for curve # CurvIndex }
10212: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10213: procedure GetPointParam
10214: {$IFDEF DELPHI}
10215: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10216: {$ELSE}
10217: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10218: { Returns the point parameters for curve # CurvIndex }
10219: procedure GetLineParam
10220: {$IFDEF DELPHI}
10221: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10222: {$ELSE}
10223: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10224: { Returns the line parameters for curve # CurvIndex }
10225: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10226: { Returns the legend for curve # CurvIndex }
10227: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10228: { Returns the step for curve # CurvIndex }
10229: {$IFDEF DELPHI}
10230: procedure PlotPoint(Canvas    : TCanvas;
```

```
10231:                     X, Y      : Float; CurvIndex : Integer); external 'dmath';
10232: {$ELSE}
10233: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10234: {$ENDIF}
10235: { Plots a point on the screen }
10236: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10237:                     X, Y                 : TVector;
10238:                    Lb, Ub, CurvIndex     : Integer); external 'dmath';
10239: { Plots a curve }
10240: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10241:                                  X, Y, S              : TVector;
10242:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10243: { Plots a curve with error bars }
10244: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10245:                    Func                 : TFunc;
10246:                    Xmin, Xmax           : Float;
10247:                    {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
10248:                    CurvIndex            : Integer); external 'dmath';
10249: { Plots a function }
10250: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10251:                       NCurv              : Integer;
10252:                       ShowPoints, ShowLines : Boolean); external 'dmath';
10253: { Writes the legends for the plotted curves }
10254: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10255:                  Nx, Ny, Nc            : Integer;
10256:                  X, Y, Z               : TVector;
10257:                  F                     : TMatrix); external 'dmath';
10258: { Contour plot }
10259: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10260: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10261: function Xuser(X : Integer):Float; external 'dmath';  {Converts screen coordinate X to user abscissa }
10262: function Yuser(Y : Integer):Float; external 'dmath';  {Converts screen coordinate Y to user ordinate }
10263: {$IFNDEF DELPHI}
10264: procedure LeaveGraphics; external 'dmath';
10265: { Quits graphic mode }
10266: {$ENDIF}
10267: { ---------------------------------------------------------------
10268:   LaTeX graphics
10269:   --------------------------------------------------------------- }
10270: function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
10271:                           Header            : Boolean) : Boolean; external 'dmath';
10272: { Initializes the LaTeX file }
10273: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10274: { Sets the graphic window }
10275: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10276: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10277: { Sets the scale on the Ox axis }
10278: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10279: { Sets the scale on the Oy axis }
10280: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10281: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10282: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10283: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10284: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10285: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10286: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10287: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10288: { Sets the maximum number of curves and re-initializes their parameters }
10289: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10290: { Sets the point parameters for curve # CurvIndex }
10291: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10292:                            Width : Float; Smooth : Boolean); external 'dmath';
10293: { Sets the line parameters for curve # CurvIndex }
10294: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10295: { Sets the legend for curve # CurvIndex }
10296: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10297: { Sets the step for curve # CurvIndex }
10298: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10299: { Plots a curve }
10300: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10301:                                      Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10302: { Plots a curve with error bars }
10303: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10304:                        Npt : Integer; CurvIndex : Integer); external 'dmath';
10305: { Plots a function }
10306: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10307: { Writes the legends for the plotted curves }
10308: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z  : TVector; F : TMatrix); external 'dmath';
10309: { Contour plot }
10310: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10311: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10312:
10313: //*************************************************unit uPSI_SynPdf;
10314:  Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10315:  Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10316:  Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10317:  Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10318:  Function PdfRect1( const Box : TPdfBox) : TPdfRect;
10319:  Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
```

```
10320:  //Function _GetCharCount( Text : PAnsiChar) : integer
10321:  //Procedure L2R( W : PWideChar; L : integer)
10322:  Function PdfCoord( MM : single) : integer
10323:  Function CurrentPrinterPaperSize : TPDFPaperSize
10324:  Function CurrentPrinterRes : TPoint
10325:  Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10326:  Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
10327:  Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10328:  Const('Usp10','String 'usp10.dll
10329:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10330:    'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10331:   AddTypeS('TScriptState_set', 'set of TScriptState_enum
10332: //*********************************************************************
10333:
10334: procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10335: begin
10336:  Procedure PMrandomize( I : word)
10337:  Function PMrandom : longint
10338:  Function Rrand : extended
10339:  Function Irand( N : word) : word
10340:  Function Brand( P : extended) : boolean
10341:  Function Nrand : extended
10342: end;
10343:
10344: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10345: begin
10346:   Function Endian( x : LongWord) : LongWord
10347:   Function Endian64( x : Int64) : Int64
10348:   Function spRol( x : LongWord; y : Byte) : LongWord
10349:   Function spRor( x : LongWord; y : Byte) : LongWord
10350:   Function Ror64( x : Int64; y : Byte) : Int64
10351: end;
10352:
10353: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10354: begin
10355:  Procedure ClearModules
10356:  Procedure ReadMapFile( Fname : string)
10357:  Function AddressInfo( Address : dword) : string
10358: end;
10359:
10360: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10361: begin
10362:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10363:    +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10364:    +'teByOther, tpExecuteByOther )
10365:   TTarPermissions', 'set of TTarPermission
10366:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10367:    +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10368:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10369:   TTarModes', 'set of TTarMode
10370:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10371:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10372:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10373:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10374:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10375:   SIRegister_TTarArchive(CL);
10376:   SIRegister_TTarWriter(CL);
10377:  Function PermissionString( Permissions : TTarPermissions) : STRING
10378:  Function ConvertFilename( Filename : STRING) : STRING
10379:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10380:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10381:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10382: end;
10383:
10384:
10385: //************************************************unit uPSI_TlHelp32;
10386: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10387: begin
10388:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10389:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10390:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10391:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10392:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10393:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10394:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10395:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10396:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10397:   AddTypeS('THeapList32', 'tagHEAPLIST32
10398:  Const('HF32_DEFAULT','LongInt'( 1);
10399:  Const('HF32_SHARED','LongInt'( 2);
10400:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10401:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10402:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10403:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10404:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10405:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10406:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10407:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10408:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
```

```
10409:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10410:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10411:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10412:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10413:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10414:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10415:    +'aPri : Longint; dwFlags : DWORD; end
10416:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10417:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10418:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10419:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10420: end;
10421:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10422:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10423:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10424:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10425:  Const('EWX_LOGOFF','LongInt'( 0);
10426:  Const('EWX_SHUTDOWN','LongInt'( 1);
10427:  Const('EWX_REBOOT','LongInt'( 2);
10428:  Const('EWX_FORCE','LongInt'( 4);
10429:  Const('EWX_POWEROFF','LongInt'( 8);
10430:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10431:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10432:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10433:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10434:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10435:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10436:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10437:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10438:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10439:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10440:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10441:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10442:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10443:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10444:  Function GetDesktopWindow : HWND
10445:  Function GetParent( hWnd : HWND) : HWND
10446:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10447:  Function GetTopWindow( hWnd : HWND) : HWND
10448:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10449:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10450:  //Delphi DFM
10451:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10452:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10453:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10454:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10455:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10456:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10457:  Function OpenIcon( hWnd : HWND) : BOOL
10458:  Function CloseWindow( hWnd : HWND) : BOOL
10459:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10460:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10461:  Function IsWindowVisible( hWnd : HWND) : BOOL
10462:  Function IsIconic( hWnd : HWND) : BOOL
10463:  Function AnyPopup : BOOL
10464:  Function BringWindowToTop( hWnd : HWND) : BOOL
10465:  Function IsZoomed( hWnd : HWND) : BOOL
10466:  Function IsWindow( hWnd : HWND) : BOOL
10467:  Function IsMenu( hMenu : HMENU) : BOOL
10468:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10469:  Function DestroyWindow( hWnd : HWND) : BOOL
10470:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10471:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10472:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10473:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10474:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10475:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10476:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10477:
10478: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10479: begin
10480:  const('ShowSetupDialogOptLong','String '--setup
10481:  PrimaryConfPathOptLong','String '--primary-config-path=
10482:  PrimaryConfPathOptShort','String '--pcp=
10483:  SecondaryConfPathOptLong','String '--secondary-config-path=
10484:  SecondaryConfPathOptShort','String '--scp=
10485:  NoSplashScreenOptLong','String '--no-splash-screen
10486:  NoSplashScreenOptShort','String '--nsc
10487:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10488:  SkipLastProjectOpt','String '--skip-last-project
10489:  DebugLogOpt','String '--debug-log=
10490:  DebugLogOptEnable','String '--debug-enable=
10491:  LanguageOpt','String '--language=
10492:  LazarusDirOpt','String '--lazarusdir=
10493:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10494:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10495:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10496:  Function IsHelpRequested : Boolean
10497:  Function IsVersionRequested : boolean
```

```
10498:  Function GetLanguageSpecified : string
10499:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10500:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10501:  Procedure ParseNoGuiCmdLineParams
10502:  Function ExtractCmdLineFilenames : TStrings
10503: end;
10504:
10505:
10506: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10507: begin
10508:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10509:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10510:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10511:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10512:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10513:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10514:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10515:  Function DirPathExists( DirectoryName : string) : boolean
10516:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10517:  Function ExtractFileNameOnly( const AFilename : string) : string
10518:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10519:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10520:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10521:  Function ForceDirectory( DirectoryName : string) : boolean
10522:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10523:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10524:  Function FileIsText( const AFilename : string) : boolean
10525:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10526:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10527:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10528:  Function TrimFilename( const AFilename : string) : string
10529:  Function ResolveDots( const AFilename : string) : string
10530:  Procedure ForcePathDelims( var FileName : string)
10531:  Function GetForcedPathDelims( const FileName : string) : String
10532:  Function CleanAndExpandFilename( const Filename : string) : string
10533:  Function CleanAndExpandDirectory( const Filename : string) : string
10534:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10535:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10536:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
      AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10537:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
      AlwaysRequireSharedBaseFolder: Boolean) : string
10538:  Function FileIsInPath( const Filename, Path : string) : boolean
10539:  Function AppendPathDelim( const Path : string) : string
10540:  Function ChompPathDelim( const Path : string) : string
10541:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10542:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10543:  Function MinimizeSearchPath( const SearchPath : string) : string
10544:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10545:  (*Function FileExistsUTF8( const Filename : string) : boolean
10546:  Function FileAgeUTF8( const FileName : string) : Longint
10547:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10548:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10549:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10550:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10551:  Procedure FindCloseUTF8( var F : TSearchrec)
10552:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10553:  Function FileGetAttrUTF8( const FileName : String) : Longint
10554:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10555:  Function DeleteFileUTF8( const FileName : String) : Boolean
10556:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10557:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10558:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10559:  Function GetCurrentDirUTF8 : String
10560:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10561:  Function CreateDirUTF8( const NewDir : String) : Boolean
10562:  Function RemoveDirUTF8( const Dir : String) : Boolean
10563:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10564:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10565:  Function FileCreateUTF8( const FileName : string) : THandle;
10566:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10567:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10568:  Function FileSizeUtf8( const Filename : string) : int64
10569:  Function GetFileDescription( const AFilename : string) : string
10570:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10571:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10572:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10573:  Function IsUNCPath( const Path : String) : Boolean
10574:  Function ExtractUNCVolume( const Path : String) : String
10575:  Function ExtractFileRoot( FileName : String) : String
10576:  Function GetDarwinSystemFilename( Filename : string) : string
10577:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10578:  Function StrToCmdLineParam( const Param : string) : string
10579:  Function MergeCmdLineParams( ParamList : TStrings) : string
10580:  Procedure InvalidateFileStateCache( const Filename : string)
10581:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10582:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10583:  Function ReadFileToString( const Filename : string) : string
10584:  type
```

```
10585:   TCopyFileFlag = ( cffOverwriteFile,
10586:     cffCreateDestDirectory, cffPreserveTime );
10587:   TCopyFileFlags = set of TCopyFileFlag;*)
10588:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10589:   TCopyFileFlags', 'set of TCopyFileFlag
10590:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10591: end;
10592:
10593: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10594: begin
10595:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10596:   SIRegister_TMask(CL);
10597:   SIRegister_TParseStringList(CL);
10598:   SIRegister_TMaskList(CL);
10599:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10600:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10601:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10602:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10603: end;
10604:
10605: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10606: begin
10607:   //PShellHookInfo', '^TShellHookInfo // will not work
10608:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10609:   SHELLHOOKINFO', 'TShellHookInfo
10610:   LPSHELLHOOKINFO', 'PShellHookInfo
10611:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10612:   SIRegister_TJvShellHook(CL);
10613:  Function InitJvShellHooks : Boolean
10614:  Procedure UnInitJvShellHooks
10615: end;
10616:
10617: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10618: begin
10619:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10620:    +', dcHasSetSel, dcWantTab, dcNative )
10621:   TDlgCodes', 'set of TDlgCode
10622:  'dcWantMessage',' dcWantAllKeys);
10623:   SIRegister_IJvExControl(CL);
10624:   SIRegister_IJvDenySubClassing(CL);
10625:   SIRegister_TStructPtrMessage(CL);
10626:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10627:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10628:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10629:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10630:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10631:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10632:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10633:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10634:  Function GetFocusedControl( AControl : TControl) : TWinControl
10635:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10636:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10637:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10638:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10639:   SIRegister_TJvExControl(CL);
10640:   SIRegister_TJvExWinControl(CL);
10641:   SIRegister_TJvExCustomControl(CL);
10642:   SIRegister_TJvExGraphicControl(CL);
10643:   SIRegister_TJvExHintWindow(CL);
10644:   SIRegister_TJvExPubGraphicControl(CL);
10645: end;
10646:
10647: (*----------------------------------------------------------------------------*)
10648: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10649: begin
10650:  Procedure EncodeStream( Input, Output : TStream)
10651:  Procedure DecodeStream( Input, Output : TStream)
10652:  Function EncodeString1( const Input : string) : string
10653:  Function DecodeString1( const Input : string) : string
10654: end;
10655:
10656: (*----------------------------------------------------------------------------*)
10657: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10658: begin
10659:   SIRegister_TWebAppRegInfo(CL);
10660:   SIRegister_TWebAppRegList(CL);
10661:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10662:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10663:  Procedure UnregisterWebApp( const AProgID : string)
10664:  Function FindRegisteredWebApp( const AProgID : string) : string
10665:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10666:  'sUDPPort','String 'UDPPort
10667: end;
10668:
10669: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10670: begin
10671:  // TStringDynArray', 'array of string
10672:  Function GetEnvVarValue( const VarName : string) : string
10673:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
```

```
10674:  Function DeleteEnvVar( const VarName : string) : Integer
10675:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
       BufSize:Int):Int;
10676:  Function ExpandEnvVars( const Str : string) : string
10677:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10678:  Procedure GetAllEnvVarNames( const Names : TStrings);
10679:  Function GetAllEnvVarNames1 : TStringDynArray;
10680:  Function EnvBlockSize : Integer
10681:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10682:   SIRegister_TPJEnvVarsEnumerator(CL);
10683:   SIRegister_TPJEnvVars(CL);
10684:   FindClass('TOBJECT'),'EPJEnvVars
10685:   FindClass('TOBJECT'),'EPJEnvVars
10686:  //Procedure Register
10687: end;
10688:
10689: (*----------------------------------------------------------------------------*)
10690: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10691: begin
10692:  'cOneSecInMS','LongInt'( 1000);
10693:  //'cDefTimeSlice','LongInt'( 50);
10694:  //'cDefMaxExecTime',' cOneMinInMS);
10695:  'cAppErrorMask','LongInt'( 1 shl 29);
10696:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10697:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10698:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10699:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10700:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10701:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10702:  Function MakeSize( const ACX, ACY : LongInt) : TSize
10703:   SIRegister_TPJCustomConsoleApp(CL);
10704:   SIRegister_TPJConsoleApp(CL);
10705: end;
10706:
10707: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10708: begin
10709:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10710:   t_encoding', '( uuencode, base64, mime )
10711:  Function internet_date( date : TDateTime) : string
10712:  Function lookup_hostname( const hostname : string) : longint
10713:  Function my_hostname : string
10714:  Function my_ip_address : longint
10715:  Function ip2string( ip_address : longint) : string
10716:  Function resolve_hostname( ip : longint) : string
10717:  Function address_from( const s : string; count : integer) : string
10718:  Function encode_base64( data : TStream) : TStringList
10719:  Function decode_base64( source : TStringList) : TMemoryStream
10720:  Function posn( const s, t : string; count : integer) : integer
10721:  Function poscn( c : char; const s : string; n : integer) : integer
10722:  Function filename_of( const s : string) : string
10723:  //Function trim( const s : string) : string
10724:  //Procedure setlength( var s : string; l : byte)
10725:  Function TimeZoneBias : longint
10726:  Function eight2seven_quoteprint( const s : string) : string
10727:  Function eight2seven_german( const s : string) : string
10728:  Function seven2eight_quoteprint( const s : string) : string end;
10729:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10730:  Function socketerror : cint
10731:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10732:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10733:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10734:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10735:  Function fplisten( s : cint; backlog : cint) : cint
10736:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10737:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10738:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10739:  Function NetAddrToStr( Entry : in_addr) : String
10740:  Function HostAddrToStr( Entry : in_addr) : String
10741:  Function StrToHostAddr( IP : String) : in_addr
10742:  Function StrToNetAddr( IP : String) : in_addr
10743:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10744:   cint8', 'shortint
10745:   cuint8', 'byte
10746:   cchar', 'cint8
10747:   cschar', 'cint8
10748:   cuchar', 'cuint8
10749:   cint16', 'smallint
10750:   cuint16', 'word
10751:   cshort', 'cint16
10752:   csshort', 'cint16
10753:   cushort', 'cuint16
10754:   cint32', 'longint
10755:   cuint32', 'longword
10756:   cint', 'cint32
10757:   csint', 'cint32
10758:   cuint', 'cuint32
10759:   csigned', 'cint
10760:   cunsigned', 'cuint
10761:   cint64', 'int64
```

```
10762:   clonglong', 'cint64
10763:   cslonglong', 'cint64
10764:   cbool', 'longbool
10765:   cfloat', 'single
10766:   cdouble', 'double
10767:   clongdouble', 'extended
10768:
10769: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10770: begin
10771:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10772:   SIRegister_TlkJSONdotnetclass(CL);
10773:   SIRegister_TlkJSONbase(CL);
10774:   SIRegister_TlkJSONnumber(CL);
10775:   SIRegister_TlkJSONstring(CL);
10776:   SIRegister_TlkJSONboolean(CL);
10777:   SIRegister_TlkJSONnull(CL);
10778:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10779:    +'se; data : TObject; var Continue : Boolean)
10780:   SIRegister_TlkJSONcustomlist(CL);
10781:   SIRegister_TlkJSONlist(CL);
10782:   SIRegister_TlkJSONobjectmethod(CL);
10783:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10784:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10785:   SIRegister_TlkHashTable(CL);
10786:   SIRegister_TlkBalTree(CL);
10787:   SIRegister_TlkJSONobject(CL);
10788:   SIRegister_TlkJSON(CL);
10789:   SIRegister_TlkJSONstreamed(CL);
10790:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10791: end;
10792:
10793: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10794: begin
10795:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10796:   SIRegister_TZSortedList(CL);
10797:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10798:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10799:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10800:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10801:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10802:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10803:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10804:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10805:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10806:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10807:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10808:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10809:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10810:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10811:  Function StrToBoolEx( Str : string) : Boolean
10812:  Function BoolToStrEx( Bool : Boolean) : String
10813:  Function IsIpAddr( const Str : string) : Boolean
10814:  Function zSplitString( const Str, Delimiters : string) : TStrings
10815:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10816:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10817:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10818:  Function FloatToSQLStr( Value : Extended) : string
10819:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10820:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10821:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10822:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10823:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10824:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10825:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10826:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10827:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10828:  Function BytesToVar( const Value : TByteDynArray) : Variant
10829:  Function VarToBytes( const Value : Variant) : TByteDynArray
10830:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10831:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10832:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10833:  Function EncodeCString( const Value : string) : string
10834:  Function DecodeCString( const Value : string) : string
10835:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10836:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10837:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
      SubVersion:Int);
10838:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
      SubVersion:Integer):Int;
10839:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10840:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10841:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10842:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10843:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10844:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10845:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10846:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10847:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10848:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
```

```
10849: end;
10850:
10851: unit uPSI_ZEncoding;
10852:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10853:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10854:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10855:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10856:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10857:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10858:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10859:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10860:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10861:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10862:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10863:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10864:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10865:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10866:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10867:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10868:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10869:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10870:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10871:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10872:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10873:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10874:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10875:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10876:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10877:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10878:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10879:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10880:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10881:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10882:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10883:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10884:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10885:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10886:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10887:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10888:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10889:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10890:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10891:  Function ZDefaultSystemCodePage : Word
10892:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10893:
10894:
10895: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10896: begin
10897:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10898:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10899:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10900:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10901:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10902:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10903:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10904:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
10905:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
10906:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
10907:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
10908:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
10909:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10910:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10911:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10912:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10913:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10914:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10915:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10916:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
10917:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
10918:  ('ilIdentiry','2 RPC_C_IMP_LEVEL_IDENTIFY);
10919:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
10920:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
10921:  ('EOAC_NONE','LongWord').SetUInt( $0);
10922:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10923:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10924:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10925:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10926:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10927:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10928:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10929:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10930:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10931:   FindClass('TOBJECT'),'EBoldCom
10932:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10933:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10934:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10935:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10936:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10937:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
```

```
10938:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10939:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10940:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10941:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10942:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10943:  Function BoldCreateGUID : TGUID
10944:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
10945:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
10946:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10947:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10948: end;
10949:
10950: (*----------------------------------------------------------------------------*)
10951: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10952: begin
10953:  Function ParseISODate( s : string) : TDateTime
10954:  Function ParseISODateTime( s : string) : TDateTime
10955:  Function ParseISOTime( str : string) : TDateTime
10956: end;
10957:
10958: (*----------------------------------------------------------------------------*)
10959: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10960: begin
10961:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10962:  Function BoldCreateGUIDWithBracketsAsString : string
10963: end;
10964:
10965: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10966: begin
10967:   FindClass('TOBJECT'),'TBoldFileHandler
10968:   FindClass('TOBJECT'),'TBoldDiskFileHandler
10969:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
10970:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
10971:   SIRegister_TBoldFileHandler(CL);
10972:   SIRegister_TBoldDiskFileHandler(CL);
10973:  Procedure BoldCloseAllFilehandlers
10974:  Procedure BoldRemoveUnchangedFilesFromEditor
10975:  Function BoldFileHandlerList : TBoldObjectArray
10976:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
       OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
10977: end;
10978:
10979: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
10980: begin
10981:   PCharArr', 'array of PChar
10982:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
10983:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
10984:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
10985:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
10986:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
10987:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
10988:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
10989:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
10990:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
10991:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
10992:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
       Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
10993:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
10994: end;
10995:
10996: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
10997: begin
10998:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
10999:   SIRegister_TfrmBoldQueryUser(CL);
11000:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
11001: end;
11002:
11003: (*----------------------------------------------------------------------------*)
11004: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
11005: begin
11006:  //('befIsInDisplayList',' BoldElementFlag0);
11007:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11008:  //('befFollowerSelected',' BoldElementFlag2);
11009:   FindClass('TOBJECT'),'TBoldQueue
11010:   FindClass('TOBJECT'),'TBoldQueueable
11011:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11012:   SIRegister_TBoldQueueable(CL);
11013:   SIRegister_TBoldQueue(CL);
11014:  Function BoldQueueFinalized : Boolean
11015:  Function BoldInstalledQueue : TBoldQueue
11016: end;
11017:
11018: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
```

```
11019: begin
11020:  const mmPerInch','Extended').setExtended( 25.4);
11021:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11022:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11023:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11024:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11025:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11026:   TBarLineType', '( white, black, black_half )
11027:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11028:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11029:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11030:   TCheckSumMethod', '( csmNone, csmModulo10 )
11031:   SIRegister_TAsBarcode(CL);
11032:  Function CheckSumModulo10( const data : string) : string
11033:  Function ConvertMmToPixelsX( const Value : Double) : Integer
11034:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11035:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11036:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11037: end;
11038:
11039: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11040: begin
11041:    THomogeneousByteVector', 'array[0..3] of Byte
11042:   THomogeneousWordVector', 'array[0..3] of Word
11043:   THomogeneousIntVector', 'array[0..3] of Integer
11044:   THomogeneousFltVector', 'array[0..3] of single
11045:   THomogeneousDblVector', 'array[0..3] of double
11046:   THomogeneousExtVector', 'array[0..3] of extended
11047:   TAffineByteVector', 'array[0..2] of Byte
11048:   TAffineWordVector', 'array[0..2] of Word
11049:   TAffineIntVector', 'array[0..2] of Integer
11050:   TAffineFltVector', 'array[0..2] of single
11051:   TAffineDblVector', 'array[0..2] of double
11052:   TAffineExtVector', 'array[0..2] of extended
11053:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11054:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11055:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11056:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11057:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11058:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11059:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11060:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11061:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11062:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11063:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11064:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11065:   TMatrix4b', 'THomogeneousByteMatrix
11066:   TMatrix4w', 'THomogeneousWordMatrix
11067:   TMatrix4i', 'THomogeneousIntMatrix
11068:   TMatrix4f', 'THomogeneousFltMatrix
11069:   TMatrix4d', 'THomogeneousDblMatrix
11070:   TMatrix4e', 'THomogeneousExtMatrix
11071:   TMatrix3b', 'TAffineByteMatrix
11072:   TMatrix3w', 'TAffineWordMatrix
11073:   TMatrix3i', 'TAffineIntMatrix
11074:   TMatrix3f', 'TAffineFltMatrix
11075:   TMatrix3d', 'TAffineDblMatrix
11076:   TMatrix3e', 'TAffineExtMatrix
11077:   //'PMatrix', '^TMatrix // will not work
11078:   TMatrixGL', 'THomogeneousFltMatrix
11079:   THomogeneousMatrix', 'THomogeneousFltMatrix
11080:   TAffineMatrix', 'TAffineFltMatrix
11081:   TQuaternion', 'record Vector : TVector4f; end
11082:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11083:   +'ger; Height : Integer; end
11084:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11085:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11086:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11087:  'EPSILON','Extended').setExtended( 1E-100);
11088:  'EPSILON2','Extended').setExtended( 1E-50);
11089:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11090:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11091:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11092:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11093:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11094:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11095:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11096:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11097:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11098:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11099:  Function VectorLength( V : array of Single) : Single
11100:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11101:  Procedure VectorNegate( V : array of Single)
11102:  Function VectorNorm( V : array of Single) : Single
11103:  Function VectorNormalize( V : array of Single) : Single
11104:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11105:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11106:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11107:  Procedure VectorScale( V : array of Single; Factor : Single)
```

```
11108:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11109:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11110:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11111:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11112:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11113:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11114:  Procedure MatrixAdjoint( var M : TMatrixGL)
11115:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11116:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11117:  Function MatrixDeterminant( M : TMatrixGL) : Single
11118:  Procedure MatrixInvert( var M : TMatrixGL)
11119:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11120:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11121:  Procedure MatrixTranspose( var M : TMatrixGL)
11122:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11123:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11124:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11125:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11126:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11127:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11128:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11129:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11130:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11131:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11132:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11133:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11134:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11135:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11136:  Function MakeAffineVector( V : array of Single) : TAffineVector
11137:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11138:  Function MakeVector( V : array of Single) : TVectorGL
11139:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11140:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11141:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11142:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11143:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11144:  Function ArcCosGL( X : Extended) : Extended
11145:  Function ArcSinGL( X : Extended) : Extended
11146:  Function ArcTan2GL( Y, X : Extended) : Extended
11147:  Function CoTanGL( X : Extended) : Extended
11148:  Function DegToRadGL( Degrees : Extended) : Extended
11149:  Function RadToDegGL( Radians : Extended) : Extended
11150:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11151:  Function TanGL( X : Extended) : Extended
11152:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11153:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11154:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11155:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11156:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11157:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11158: end;
11159:
11160:
11161: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11162: begin
11163:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11164:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11165:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11166:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11167:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11168:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11169:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11170:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11171:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11172:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11173:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11174:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11175:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11176:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11177:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11178:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11179:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11180:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11181:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11182:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11183:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11184:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11185:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11186:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11187:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
      Items:TStrings):Bool;
11188:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
      SaveTo:TStrings):Bool;
11189:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11190: end;
11191:
11192: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11193: begin
11194:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
```

```
11195:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11196:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11197:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11198:   FindClass('TOBJECT'),'EInvalidParam
11199:  Function IsDCOMInstalled : Boolean
11200:  Function IsDCOMEnabled : Boolean
11201:  Function GetDCOMVersion : string
11202:  Function GetMDACVersion : string
11203:  Function GetMDACVersion2 : string
11204:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11205:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11206:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11207:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11208:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11209:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11210:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11211:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11212:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11213:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11214:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11215:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11216:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11217:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11218: end;
11219:
11220:
11221: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11222: begin
11223:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11224:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11225:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11226:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11227:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11228:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11229:  DegPerCycle','Extended').setExtended( 360.0);
11230:  DegPerGrad','Extended').setExtended( 0.9);
11231:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11232:  GradPerCycle','Extended').setExtended( 400.0);
11233:  GradPerDeg','Extended').setExtended( 1.1111111111111111111111111111111);
11234:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11235:  RadPerCycle','Extended').setExtended( 6.2831853071795864769252867665590);
11236:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11237:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11238:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11239:  CyclePerGrad','Extended').setExtended( 0.0025);
11240:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11241:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11242:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11243:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11244:  Function MakePercentage( const Step, Max : Longint) : Longint
11245:  Function CelsiusToKelvin( const T : double) : double
11246:  Function CelsiusToFahrenheit( const T : double) : double
11247:  Function KelvinToCelsius( const T : double) : double
11248:  Function KelvinToFahrenheit( const T : double) : double
11249:  Function FahrenheitToCelsius( const T : double) : double
11250:  Function FahrenheitToKelvin( const T : double) : double
11251:  Function CycleToDeg( const Cycles : double) : double
11252:  Function CycleToGrad( const Cycles : double) : double
11253:  Function CycleToRad( const Cycles : double) : double
11254:  Function DegToCycle( const Degrees : double) : double
11255:  Function DegToGrad( const Degrees : double) : double
11256:  Function DegToRad( const Degrees : double) : double
11257:  Function GradToCycle( const Grads : double) : double
11258:  Function GradToDeg( const Grads : double) : double
11259:  Function GradToRad( const Grads : double) : double
11260:  Function RadToCycle( const Radians : double) : double
11261:  Function RadToDeg( const Radians : double) : double
11262:  Function RadToGrad( const Radians : double) : double
11263:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11264:  Function DmsToRad( const D, M : Integer; const S : double) : double
11265:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11266:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11267:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11268:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11269:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11270:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11271:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11272:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11273:  Function CmToInch( const Cm : double) : double
11274:  Function InchToCm( const Inch : double) : double
11275:  Function FeetToMetre( const Feet : double) : double
11276:  Function MetreToFeet( const Metre : double) : double
11277:  Function YardToMetre( const Yard : double) : double
11278:  Function MetreToYard( const Metre : double) : double
11279:  Function NmToKm( const Nm : double) : double
11280:  Function KmToNm( const Km : double) : double
```

```
11281:  Function KmToSm( const Km : double) : double
11282:  Function SmToKm( const Sm : double) : double
11283:  Function LitreToGalUs( const Litre : double) : double
11284:  Function GalUsToLitre( const GalUs : double) : double
11285:  Function GalUsToGalCan( const GalUs : double) : double
11286:  Function GalCanToGalUs( const GalCan : double) : double
11287:  Function GalUsToGalUk( const GalUs : double) : double
11288:  Function GalUkToGalUs( const GalUk : double) : double
11289:  Function LitreToGalCan( const Litre : double) : double
11290:  Function GalCanToLitre( const GalCan : double) : double
11291:  Function LitreToGalUk( const Litre : double) : double
11292:  Function GalUkToLitre( const GalUk : double) : double
11293:  Function KgToLb( const Kg : double) : double
11294:  Function LbToKg( const Lb : double) : double
11295:  Function KgToOz( const Kg : double) : double
11296:  Function OzToKg( const Oz : double) : double
11297:  Function CwtUsToKg( const Cwt : double) : double
11298:  Function CwtUkToKg( const Cwt : double) : double
11299:  Function KaratToKg( const Karat : double) : double
11300:  Function KgToCwtUs( const Kg : double) : double
11301:  Function KgToCwtUk( const Kg : double) : double
11302:  Function KgToKarat( const Kg : double) : double
11303:  Function KgToSton( const Kg : double) : double
11304:  Function KgToLton( const Kg : double) : double
11305:  Function StonToKg( const STon : double) : double
11306:  Function LtonToKg( const Lton : double) : double
11307:  Function QrUsToKg( const Qr : double) : double
11308:  Function QrUkToKg( const Qr : double) : double
11309:  Function KgToQrUs( const Kg : double) : double
11310:  Function KgToQrUk( const Kg : double) : double
11311:  Function PascalToBar( const Pa : double) : double
11312:  Function PascalToAt( const Pa : double) : double
11313:  Function PascalToTorr( const Pa : double) : double
11314:  Function BarToPascal( const Bar : double) : double
11315:  Function AtToPascal( const At : double) : double
11316:  Function TorrToPascal( const Torr : double) : double
11317:  Function KnotToMs( const Knot : double) : double
11318:  Function HpElectricToWatt( const HpE : double) : double
11319:  Function HpMetricToWatt( const HpM : double) : double
11320:  Function MsToKnot( const ms : double) : double
11321:  Function WattToHpElectric( const W : double) : double
11322:  Function WattToHpMetric( const W : double) : double
11323:  function getBigPI: string;    //PI of 1000 numbers
11324:
11325: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11326: begin
11327:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11328:  Procedure CDCopyFile( const FileName, DestName : string)
11329:  Procedure CDMoveFile( const FileName, DestName : string)
11330:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11331:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11332:  Function CDGetTempDir : string
11333:  Function CDGetFileSize( FileName : string) : longint
11334:  Function GetFileTime( FileName : string) : longint
11335:  Function GetShortName( FileName : string) : string
11336:  Function GetFullName( FileName : string) : string
11337:  Function WinReboot : boolean
11338:  Function WinDir : String
11339:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11340:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11341:  Function devExecutor : TdevExecutor
11342: end;
11343:
11344: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11345: begin
11346:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11347:  Procedure Associate( Index : integer)
11348:  Procedure UnAssociate( Index : integer)
11349:  Function IsAssociated( Index : integer) : boolean
11350:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11351:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11352:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11353:  procedure RefreshIcons;
11354: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11355: function MergColor(Colors: Array of TColor): TColor;
11356: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11357: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11358: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11359: function GetInverseColor(AColor: TColor): TColor;
11360: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11361: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11362: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11363: Procedure GetSystemMenuFont(Font: TFont);
11364: end;
11365:
11366: //************************unit uPSI_JvHLParser;*****************************
11367: function IsStringConstant(const St: string): Boolean;
11368: function IsIntConstant(const St: string): Boolean;
11369: function IsRealConstant(const St: string): Boolean;
```

```
11370: function IsIdentifier(const ID: string): Boolean;
11371: function GetStringValue(const St: string): string;
11372: procedure ParseString(const S: string; Ss: TStrings);
11373: function IsStringConstantW(const St: WideString): Boolean;
11374: function IsIntConstantW(const St: WideString): Boolean;
11375: function IsRealConstantW(const St: WideString): Boolean;
11376: function IsIdentifierW(const ID: WideString): Boolean;
11377: function GetStringValueW(const St: WideString): WideString;
11378: procedure ParseStringW(const S: WideString; Ss: TStrings);
11379:
11380:
11381: //*************************unit uPSI_JclMapi;*****************************
11382:
11383: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11384: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11385: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11386: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11387: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11388:
11389: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11390: begin
11391:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11392:  Function BuildType1Message( ADomain, AHost : String) : String
11393:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11394:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11395:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11396: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11397: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11398: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11399: end;
11400:
11401: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11402: begin
11403: ('IpAny','LongWord').SetUInt( $00000000);
11404:  IpLoopBack','LongWord').SetUInt( $7F000001);
11405:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11406:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11407:  PortAny','LongWord( $0000);
11408:  SocketMaxConnections','LongInt'( 5);
11409:  TIpAddr', 'LongWord
11410:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11411:  Function HostToNetLong( HostLong : LongWord) : LongWord
11412:  Function HostToNetShort( HostShort : Word) : Word
11413:  Function NetToHostLong( NetLong : LongWord) : LongWord
11414:  Function NetToHostShort( NetShort : Word) : Word
11415:  Function StrToIp( Ip : string) : TIpAddr
11416:  Function IpToStr( Ip : TIpAddr) : string
11417: end;
11418:
11419: (*----------------------------------------------------------------------------*)
11420: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11421: begin
11422:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11423:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11424:    +'amSha1, AlsmtpClientAuthAutoSelect )
11425:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11426:   SIRegister_TAlSmtpClient(CL);
11427: end;
11428:
11429: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11430: begin
11431: 'TBitNo', 'Integer
11432:   TStByteNo', 'Integer
11433: TStationNo', 'Integer
11434: TInOutNo', 'Integer
11435: TIo', '( EE, AA, NE, NA )
11436: TBitSet', 'set of TBitNo
11437: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11438: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11439: TBitAddr', 'LongInt
11440: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11441: TByteAddr', 'SmallInt
11442: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11443:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11444:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11445:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
       aBitNo:TBitNo);
11446:  Function BitAddrToStr( Value : TBitAddr) : string
11447:  Function StrToBitAddr( const Value : string) : TBitAddr
11448:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11449:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11450:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11451:  Function ByteAddrToStr( Value : TByteAddr) : string
11452:  Function StrToByteAddr( const Value : string) : TByteAddr
11453:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11454:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
```

```
11455:  Function InOutStateToStr( State : TInOutState) : string
11456:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11457:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11458: end;
11459:
11460: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11461: begin
11462: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11463:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11464: DpmiPmVector', 'Int64
11465:  'DInterval','LongInt'( 1000);
11466:  //'DEnabled','Boolean')BoolToStr( True);
11467:  'DIntFreq','string' if64
11468:  //'DMessages','Boolean if64);
11469:   SIRegister_TwdxCustomTimer(CL);
11470:   SIRegister_TwdxTimer(CL);
11471:   SIRegister_TwdxRtcTimer(CL);
11472:   SIRegister_TCustomIntTimer(CL);
11473:   SIRegister_TIntTimer(CL);
11474:   SIRegister_TRtcIntTimer(CL);
11475:  Function RealNow : TDateTime
11476:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11477:  Function DateTimeToMs( Time : TDateTime) : LongInt
11478: end;
11479:
11480: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11481: begin
11482: TIdSyslogPRI', 'Integer
11483: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11484:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11485:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11486:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11487:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11488: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11489:   SIRegister_TIdSysLogMsgPart(CL);
11490:   SIRegister_TIdSysLogMessage(CL);
11491:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11492:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11493:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11494:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11495:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11496:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11497: end;
11498:
11499: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11500: begin
11501:  'UWhitespace','String '(?:\s*)
11502:  Function StripSpaces( const AText : string) : string
11503:  Function CharCount( const AText : string; Ch : Char) : Integer
11504:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11505:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11506: end;
11507:
11508:
11509: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11510: begin
11511:  ExtPascalVersion','String '0.9.8
11512:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11513:    +'Opera, brKonqueror, brMobileSafari )
11514:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11515:   AddTypeS('TExtProcedure', 'Procedure
11516:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11517:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11518:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11519:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11520:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11521:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11522:  Function StrToJS( const S : string; UseBR : boolean) : string
11523:  Function CaseOf( const S : string; const Cases : array of string) : integer
11524:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11525:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11526:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11527:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11528:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11529:  Function IsUpperCase( S : string) : boolean
11530:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11531:  Function BeautifyCSS( const AStyle : string) : string
11532:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11533:  Function JSDateToDateTime( JSDate : string) : TDateTime
11534: end;
11535:
11536: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11537: begin
11538:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11539:   TSHDeleteOptions', 'set of TSHDeleteOption
11540:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11541:   TSHRenameOptions', 'set of TSHRenameOption
11542:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11543:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
```

```
11544:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11545:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11546:   TEnumFolderFlags', 'set of TEnumFolderFlag
11547:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11548:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11549:    +'IEnumIdList; Folder : IShellFolder; end
11550:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11551:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11552:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11553:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11554:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11555:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11556:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11557:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11558:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11559:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11560:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11561:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11562:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11563:  Function SHFreeMem( var P : Pointer) : Boolean
11564:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11565:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11566:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11567:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11568:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11569:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11570:  Function PidlFree( var IdList : PItemIdList) : Boolean
11571:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11572:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11573:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11574:  Function PidlToPath( IdList : PItemIdList) : string
11575:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11576:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11577:   PShellLink', '^TShellLink // will not work
11578:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11579:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11580:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11581:  Procedure ShellLinkFree( var Link : TShellLink)
11582:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11583:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11584:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11585:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11586:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11587:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11588:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11589:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11590:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11591:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11592:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11593:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11594:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11595:  Function ShellOpenAs( const FileName : string) : Boolean
11596:  Function ShellRasDial( const EntryName : string) : Boolean
11597:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11598:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11599:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11600:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11601:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11602:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11603:  Function OemKeyScan( wOemChar : Word) : DWORD
11604:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11605: end;
11606:
11607: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11608: begin
11609:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11610:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11611:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11612:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11613:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11614:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11615:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11616:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11617:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11618:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11619:  Function xmlValidName( const Text : UnicodeString) : Boolean
11620:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11621:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11622:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11623:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11624:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
      : TUnicodeCodecClass
11625:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11626:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11627:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11628:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11629:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11630:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11631:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
```

```
11632:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11633:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11634:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11635:  Procedure SelfTestcXMLFunctions
11636: end;
11637:
11638: (*----------------------------------------------------------------------------*)
11639: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11640: begin
11641:  Function AWaitCursor : IUnknown
11642:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11643:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11644:  Function YesNo( const ACaption, AMsg : string) : boolean
11645:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11646:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11647:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11648:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11649:  Procedure GetSystemPaths( Strings : TStrings)
11650:  Procedure MakeEditNumeric( EditHandle : integer)
11651: end;
11652:
11653: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11654: begin
11655:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11656:  'BI_YUY2','LongWord( $32595559);
11657:  'BI_UYVY','LongWord').SetUInt( $59565955);
11658:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11659:  'BI_YVU9','LongWord').SetUInt( $39555659);
11660:  'BI_YUV12','LongWord( $30323449);
11661:  'BI_Y8','LongWord').SetUInt( $20203859);
11662:  'BI_Y211','LongWord').SetUInt( $31313259);
11663:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11664:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11665: end;
11666:
11667: (*----------------------------------------------------------------------------*)
11668: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11669: begin
11670:  'WM_USER','LongWord').SetUInt( $0400);
11671:  'WM_CAP_START','LongWord').SetUint($0400);
11672:  'WM_CAP_END','longword').SetUint($0400+85);
11673:  //WM_CAP_START+  85
11674:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11675:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11676:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11677:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11678:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11679:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11680:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11681:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11682:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11683:  Function capGetUserData( hwnd : THandle) : LongInt
11684:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11685:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11686:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11687:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11688:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11689:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11690:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11691:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11692:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11693:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11694:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11695:  Function capEditCopy( hwnd : THandle) : LongInt
11696:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11697:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11698:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11699:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11700:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11701:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11702:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11703:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11704:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11705:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11706:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11707:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11708:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11709:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11710:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11711:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11712:  Function capGrabFrame( hwnd : THandle) : LongInt
11713:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11714:  Function capCaptureSequence( hwnd : THandle) : LongInt
11715:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11716:  Function capCaptureStop( hwnd : THandle) : LongInt
11717:  Function capCaptureAbort( hwnd : THandle) : LongInt
11718:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11719:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11720:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
```

```
11721:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11722:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11723:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11724:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11725:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11726:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11727:  Function capPalettePaste( hwnd : THandle) : LongInt
11728:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11729:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11730:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11731:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11732:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11733:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11734:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11735:   //PCapStatus', '^TCapStatus // will not work
11736:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11737:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11738:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11739:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11740:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11741:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11742:    +' wNumAudioAllocated : WORD; end
11743:   //PCaptureParms', '^TCaptureParms // will not work
11744:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11745:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11746:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11747:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11748:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11749:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11750:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11751:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11752:    +'he : BOOL; AVStreamMaster : WORD; end
11753:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11754:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11755:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11756:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11757:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
      : Integer; hwndParent : THandle; nID : Integer) : THandle
11758:  Function
      capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11759:  'IDS_CAP_BEGIN','LongInt'( 300);
11760:  'IDS_CAP_END','LongInt'( 301);
11761:  'IDS_CAP_INFO','LongInt'( 401);
11762:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11763:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11764:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11765:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11766:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11767:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11768:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11769:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11770:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11771:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11772:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11773:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11774:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11775:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11776:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11777:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11778:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11779:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11780:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11781:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11782:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11783:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11784:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11785:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11786:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11787:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11788:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11789:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11790:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11791:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11792:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11793:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11794:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11795:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11796:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11797:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11798:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11799:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11800:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11801:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11802:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11803:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11804:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11805:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11806:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11807:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
```

```
11808:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11809:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11810:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11811:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11812:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11813:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11814:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11815:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11816:  'AVICAP32','String 'AVICAP32.dll
11817: end;
11818:
11819: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11820: begin
11821:  Function AlBoolToInt( Value : Boolean) : Integer
11822:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11823:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11824:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11825:  Function ALInc( var x : integer; Count : integer) : Integer
11826:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11827:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11828:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11829:  Function ALIsInteger(const S: AnsiString): Boolean;
11830:  function ALIsDecimal(const S: AnsiString): boolean;
11831:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11832:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11833:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11834:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11835:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11836:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11837:  Function ALRandomStr(const aLength: Longint): AnsiString;
11838:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11839:  Function ALRandomStrU(const aLength: Longint): String;
11840: end;
11841:
11842: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11843: begin
11844:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
      aTrueStr: AnsiString; const aFalseStr : AnsiString)
11845: end;
11846:
11847: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11848: begin
11849:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11850:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11851:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11852:    +'; ullAvailExtendedVirtual : Int64; end
11853:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11854:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11855:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11856:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11857:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11858:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11859:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11860:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11861: end;
11862:
11863: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11864: begin
11865:   SIRegister_THandledObject(CL);
11866:   SIRegister_TEvent(CL);
11867:   SIRegister_TMutex(CL);
11868:   SIRegister_TSharedMem(CL);
11869:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11870:  'TRACE_BUFFER','String 'TRACE_BUFFER
11871:  'TRACE_MUTEX','String 'TRACE_MUTEX
11872:   //PTraceEntry', '^TTraceEntry // will not work
11873:   SIRegister_TIPCTracer(CL);
11874:  'MAX_CLIENTS','LongInt'( 6);
11875:  'IPCTIMEOUT','LongInt'( 2000);
11876:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11877:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11878:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11879:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11880:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11881:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11882:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11883:   FindClass('TOBJECT'),'EMonitorActive
11884:   FindClass('TOBJECT'),'TIPCThread
11885:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11886:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11887:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11888:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11889:   TClientFlags', 'set of TClientFlag
11890:   //PEventData', '^TEventData // will not work
11891:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11892:    +'lag; Flags : TClientFlags; end
11893:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11894:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11895:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
```

```
11896:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11897:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11898:   SIRegister_TIPCEvent(CL);
11899:   //PClientDirRecords', '^TClientDirRecords // will not work
11900:   SIRegister_TClientDirectory(CL);
11901:   TIPCState', '( stInActive, stDisconnected, stConnected )
11902:   SIRegister_TIPCThread(CL);
11903:   SIRegister_TIPCMonitor(CL);
11904:   SIRegister_TIPCClient(CL);
11905:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11906: end;
11907:
11908: (*---------------------------------------------------------------------------*)
11909: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11910: begin
11911:   SIRegister_TAlGSMComm(CL);
11912:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11913:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
      AMessage:AnsiString);
11914:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11915:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
      UseGreekAlphabet:Bool):Widestring;
11916:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11917:  end;
11918:
11919: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11920: begin
11921:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11922:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11923:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11924:   TInternetScheme', 'integer
11925:   TALIPv6Binary', 'array[1..16] of Char;
11926:  // TALIPv6Binary = array[1..16] of ansiChar;
11927:  //   TInternetScheme = Integer;
11928:   SIRegister_TALHTTPRequestHeader(CL);
11929:   SIRegister_TALHTTPCookie(CL);
11930:   SIRegister_TALHTTPCookieCollection(CL);
11931:   SIRegister_TALHTTPResponseHeader(CL);
11932:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11933:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11934: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
      Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11935: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
      Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11936: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
      Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11937:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11938:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11939:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11940:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11941:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11942:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
      ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11943:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
      Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11944:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11945:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11946:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11947:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11948:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11949:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11950:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11951:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11952:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11953:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11954:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
11955:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11956:  Function ALZeroIpV6 : TALIPv6Binary
11957:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11958:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
11959:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
11960:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
11961: end;
11962:
11963: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11964: begin
11965:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
      DecodeHTMLText:Bool;
11966:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11967:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
11968:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
11969:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
11970:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
      useNumRef:bool):AnsiString);
11971:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
11972:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
11973:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
11974:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
      DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
```

```
11975:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
11976: end;
11977:
11978: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11979: begin
11980:   SIRegister_TALEMailHeader(CL);
11981:   SIRegister_TALNewsArticleHeader(CL);
11982:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
11983:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
11984:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
11985:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
11986:  Function AlGenerateInternetMessageID : AnsiString;
11987:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
11988:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
11989:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
11990: end;
11991:
11992: (*----------------------------------------------------------------------------*)
11993: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
11994: begin
11995:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
11996:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
11997:  Function ALgetLocalIPs : TALStrings
11998:  Function ALgetLocalHostName : AnsiString
11999: end;
12000:
12001: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12002: begin
12003:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
12004:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12005:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12006:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
12007:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12008:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12009:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
12010:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12011:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
12012: end;
12013:
12014: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12015: begin
12016:   TStartupInfoA', 'TStartupInfo
12017:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12018:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12019:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12020:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12021:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12022:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12023:  SE_TCB_NAME','String 'SeTcbPrivilege
12024:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12025:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12026:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12027:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12028:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12029:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12030:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12031:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12032:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12033:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12034:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12035:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12036:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12037:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12038:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12039:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12040:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12041:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12042:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12043:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12044:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12045:  Function AlGetEnvironmentString : AnsiString
12046:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12047:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12048:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12049:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12050:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12051: end;
```

```
12052:
12053: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12054: begin
12055:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12056:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12057:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12058:  Function ALGetModuleName : ansistring
12059:  Function ALGetModuleFileNameWithoutExtension : ansistring
12060:  Function ALGetModulePath : ansiString
12061:  Function AlGetFileSize( const AFileName : ansistring) : int64
12062:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12063:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12064:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12065:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12066:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12067:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12068:  Function ALFileExists( const Path : ansiString) : boolean
12069:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12070:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12071:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12072:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12073:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12074: end;
12075:
12076: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12077: begin
12078:   NativeInt', 'Integer
12079:   NativeUInt', 'Cardinal
12080:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12081:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12082:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12083:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12084:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12085:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12086:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12087:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12088:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12089:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12090:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12091:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12092:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12093:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12094:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12095:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12096:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12097:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12098:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
12099:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12100:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12101:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12102:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12103:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12104:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12105:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12106:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12107:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12108:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12109:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12110:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12111:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12112: end;
12113:
12114: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12115: begin
12116:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12117:   FindClass('TOBJECT'),'TALXMLNode
12118:   FindClass('TOBJECT'),'TALXMLNodeList
12119:   FindClass('TOBJECT'),'TALXMLDocument
12120:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12121:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12122:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12123:    +'nst Name : AnsiString; const Attributes : TALStrings)
12124:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12125:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
```

```
12126:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12127:    +'ntDocType, ntDocFragment, ntNotation )
12128:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12129:   TALXMLDocOptions', 'set of TALXMLDocOption
12130:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12131:   TALXMLParseOptions', 'set of TALXMLParseOption
12132:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12133:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12134:   SIRegister_EALXMLDocError(CL);
12135:   SIRegister_TALXMLNodeList(CL);
12136:   SIRegister_TALXMLNode(CL);
12137:   SIRegister_TALXmlElementNode(CL);
12138:   SIRegister_TALXmlAttributeNode(CL);
12139:   SIRegister_TALXmlTextNode(CL);
12140:   SIRegister_TALXmlDocumentNode(CL);
12141:   SIRegister_TALXmlCommentNode(CL);
12142:   SIRegister_TALXmlProcessingInstrNode(CL);
12143:   SIRegister_TALXmlCDataNode(CL);
12144:   SIRegister_TALXmlEntityRefNode(CL);
12145:   SIRegister_TALXmlEntityNode(CL);
12146:   SIRegister_TALXmlDocTypeNode(CL);
12147:   SIRegister_TALXmlDocFragmentNode(CL);
12148:   SIRegister_TALXmlNotationNode(CL);
12149:   SIRegister_TALXMLDocument(CL);
12150:  cALXMLUTF8EncodingStr','String 'UTF-8
12151:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12152:  CALNSDelim','String ':
12153:  CALXML','String 'xml
12154:  CALVersion','String 'version
12155:  CALEncoding','String 'encoding
12156:  CALStandalone','String 'standalone
12157:  CALDefaultNodeIndent','String '
12158:  CALXmlDocument','String 'DOCUMENT
12159:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12160:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
      EncodingStr:AnsiString);
12161:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
      ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12162:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
      ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12163:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
      Recurse: Boolean):TalxmlNode
12164:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
      AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12165:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
      AnsiString
12166: end;
12167:
12168: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12169: //based on TEEProc, TeCanvas, TEEngine, TChart
12170: begin
12171:  'TeePiStep','Double').setExtended( Pi / 180.0);
12172:  'TeeDefaultPerspective','LongInt'( 100);
12173:  'TeeMinAngle','LongInt'( 270);
12174:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12175:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12176:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12177:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12178:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12179:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12180:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12181:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12182:  'TA_LEFT','LongInt'( 0);
12183:  'TA_RIGHT','LongInt'( 2);
12184:  'TA_CENTER','LongInt'( 6);
12185:  'TA_TOP','LongInt'( 0);
12186:  'TA_BOTTOM','LongInt'( 8);
12187:  'teePATCOPY','LongInt'( 0);
12188:  'NumCirclePoints','LongInt'( 64);
12189:  'teeDEFAULT_CHARSET','LongInt'( 1);
12190:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12191:  'TA_LEFT','LongInt'( 0);
12192:  'bs_Solid','LongInt'( 0);
12193:  'teepf24Bit','LongInt'( 0);
12194:  'teepfDevice','LongInt'( 1);
12195:  'CM_MOUSELEAVE','LongInt'( 10000);
12196:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12197:  'DC_BRUSH','LongInt'( 18);
12198:  'DC_PEN','LongInt'( 19);
12199:   teeCOLORREF', 'LongWord
12200:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12201:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12202:   SIRegister_TFilterRegion(CL);
12203:   SIRegister_IFormCreator(CL);
12204:   SIRegister_TTeeFilter(CL);
12205:   //TFilterClass', 'class of TTeeFilter
12206:   SIRegister_TFilterItems(CL);
12207:   SIRegister_TConvolveFilter(CL);
12208:   SIRegister_TBlurFilter(CL);
```

```
12209:   SIRegister_TTeePicture(CL);
12210:   TPenEndStyle', '( esRound, esSquare, esFlat )
12211:   SIRegister_TChartPen(CL);
12212:   SIRegister_TChartHiddenPen(CL);
12213:   SIRegister_TDottedGrayPen(CL);
12214:   SIRegister_TDarkGrayPen(CL);
12215:   SIRegister_TWhitePen(CL);
12216:   SIRegister_TChartBrush(CL);
12217:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12218:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12219:   SIRegister_TView3DOptions(CL);
12220:   FindClass('TOBJECT'),'TTeeCanvas
12221:   TTeeTransparency', 'Integer
12222:   SIRegister_TTeeBlend(CL);
12223:   FindClass('TOBJECT'),'TCanvas3D
12224:   SIRegister_TTeeShadow(CL);
12225:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
      gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12226:   FindClass('TOBJECT'),'TSubGradient
12227:   SIRegister_TCustomTeeGradient(CL);
12228:   SIRegister_TSubGradient(CL);
12229:   SIRegister_TTeeGradient(CL);
12230:   SIRegister_TTeeFontGradient(CL);
12231:   SIRegister_TTeeFont(CL);
12232:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12233:   TCanvasTextAlign', 'Integer
12234:   TTeeCanvasHandle', 'HDC
12235:   SIRegister_TTeeCanvas(CL);
12236:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12237:   SIRegister_TFloatXYZ(CL);
12238:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
12239:   TRGB', 'record blue: byte; green: byte; red: byte; end
12240:   {TRGB=packed record
12241:     Blue  : Byte;
12242:     Green : Byte;
12243:     Red   : Byte;
12244:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12245:
12246: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12247:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12248: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12249: TCanvas3DPlane', '( cpX, cpY, cpZ )
12250:   //IInterface', 'IUnknown
12251:   SIRegister_TCanvas3D(CL);
12252:   SIRegister_TTeeCanvas3D(CL);
12253:   TTrianglePoints', 'Array[0..2] of TPoint;
12254:   TFourPoints', 'Array[0..3] of TPoint;
12255:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12256:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12257:  Function Point3D( const x, y, z : Integer) : TPoint3D
12258:  Procedure SwapDouble( var a, b : Double)
12259:  Procedure SwapInteger( var a, b : Integer)
12260:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12261:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12262:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12263:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12264:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12265:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12266:  Procedure UnClipCanvas( ACanvas : TCanvas)
12267:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12268:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12269:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12270:  'TeeCharForHeight','String 'W
12271:  'DarkerColorQuantity','Byte').SetUInt( 128);
12272:  'DarkColorQuantity','Byte').SetUInt( 64);
12273:   TButtonGetColorProc', 'Function  : TColor
12274:   SIRegister_TTeeButton(CL);
12275:   SIRegister_TButtonColor(CL);
12276:   SIRegister_TComboFlat(CL);
12277:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12278:  Function TeePoint( const aX, aY : Integer) : TPoint
12279:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12280:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12281:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12282:  Function OrientRectangle( const R : TRect) : TRect
12283:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12284:  Function PolygonBounds( const P : array of TPoint) : TRect
12285:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12286:  Function RGBValue( const Color : TColor) : TRGB
12287:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12288:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12289:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12290:  Function TeeCull( const P : TFourPoints) : Boolean;
12291:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12292:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12293:  Procedure SmoothStretch( Src, Dst : TBitmap);
12294:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12295:  Function TeeDistance( const x, y : Double) : Double
12296:  Function TeeLoadLibrary( const FileName : String) : HInst
```

```
12297:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12298:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12299:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12300:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
       Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12301:    SIRegister_ICanvasHyperlinks(CL);
12302:    SIRegister_ICanvasToolTips(CL);
12303:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12304: end;
12305:
12306: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12307: begin
12308:    TOvcHdc', 'Integer
12309:    TOvcHWND', 'Cardinal
12310:    TOvcHdc', 'HDC
12311:    TOvcHWND', 'HWND
12312:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12313:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12314:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12315:  Function DefaultEpoch : Integer
12316:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12317:  Procedure FixRealPrim( P : PChar; DC : Char)
12318:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12319:  Function GetLeftButton : Byte
12320:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12321:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12322:  Function GetShiftFlags : Byte
12323:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12324:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12325:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12326:  Function ovIsForegroundTask : Boolean
12327:  Function ovTrimLeft( const S : string) : string
12328:  Function ovTrimRight( const S : string) : string
12329:  Function ovQuotedStr( const S : string) : string
12330:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12331:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12332:  Function PtrDiff( const P1, P2 : PChar) : Word
12333:  Procedure PtrInc( var P, Delta : Word)
12334:  Procedure PtrDec( var P, Delta : Word)
12335:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12336:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
       SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12337:  Function ovMinI( X, Y : Integer) : Integer
12338:  Function ovMaxI( X, Y : Integer) : Integer
12339:  Function ovMinL( X, Y : LongInt) : LongInt
12340:  Function ovMaxL( X, Y : LongInt) : LongInt
12341:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12342:  Function PartialCompare( const S1, S2 : string) : Boolean
12343:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12344:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12345:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12346:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
       TransparentColor : TColor)
12347:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
       TransparentColor : TColorRef)
12348:  Function ovWidthOf( const R : TRect) : Integer
12349:  Function ovHeightOf( const R : TRect) : Integer
12350:  Procedure ovDebugOutput( const S : string)
12351:  Function GetArrowWidth( Width, Height : Integer) : Integer
12352:  Procedure StripCharSeq( CharSeq : string; var Str : string)
12353:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12354:  Procedure StripCharFromFront( aChr : Char; var Str : string)
12355:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12356:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12357:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12358:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12359:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12360:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12361:  Function CreateMetaFile( p1 : PChar) : HDC
12362:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12363:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12364:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12365:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12366:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12367:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12368:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12369:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12370:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12371:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12372:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12373:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12374:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
       SrcHeight:Int;Rop:DWORD):BOOL
12375:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12376:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
       SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12377:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12378:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12379:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
```

```
12380:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12381:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12382:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12383:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12384:  Function UpdateColors( DC : HDC) : BOOL
12385:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12386:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12387:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12388:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12389:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12390:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12391:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
       HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12392:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
       yMask:Int):BOOL;
12393:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12394:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12395:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12396:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12397:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12398:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12399:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12400:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12401:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12402:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12403:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12404:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12405: end;
12406:
12407: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12408: begin
12409:   SIRegister_TOvcAbstractStore(CL);
12410:   //PExPropInfo', '^TExPropInfo // will not work
12411: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12412:   SIRegister_TOvcPropertyList(CL);
12413:   SIRegister_TOvcDataFiler(CL);
12414:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12415:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12416:  Function CreateStoredItem( const CompName, PropName : string) : string
12417:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12418:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12419: end;
12420:
12421: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12422: begin
12423:  'ovsetsize','LongInt'( 16);
12424:  'etSyntax','LongInt'( 0);
12425:  'etSymantic','LongInt'( 1);
12426:  'chCR','Char #13);
12427:  'chLF','Char #10);
12428:  'chLineSeparator',' chCR);
12429:   SIRegister_TCocoError(CL);
12430:   SIRegister_TCommentItem(CL);
12431:   SIRegister_TCommentList(CL);
12432:   SIRegister_TSymbolPosition(CL);
12433: TGenListType', '( glNever, glAlways, glOnError )
12434: TovBitSet', 'set of Integer
12435:   //PStartTable', '^TStartTable // will not work
12436: 'TovCharSet', 'set of AnsiChar
12437: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12438: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12439: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12440: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12441: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12442:    +'osition; const Data : string; ErrorType : integer)
12443: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12444: TGetCH', 'Function ( pos : longint) : char
12445: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12446:   SIRegister_TCocoRScanner(CL);
12447:   SIRegister_TCocoRGrammar(CL);
12448:  '_EF','Char #0);
12449:  '_TAB','Char').SetString( #09);
12450:  '_CR','Char').SetString( #13);
12451:  '_LF','Char').SetString( #10);
12452:  '_EL','').SetString( _CR);
12453:  '_EOF','Char').SetString( #26);
12454:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12455:  'minErrDist','LongInt'( 2);
12456:  Function ovPadL( S : string; ch : char; L : integer) : string
12457: end;
12458:
12459:   TFormatSettings = record
12460:     CurrencyFormat: Byte;
12461:     NegCurrFormat: Byte;
12462:     ThousandSeparator: Char;
12463:     DecimalSeparator: Char;
12464:     CurrencyDecimals: Byte;
12465:     DateSeparator: Char;
12466:     TimeSeparator: Char;
```

```
12467:     ListSeparator: Char;
12468:     CurrencyString: string;
12469:     ShortDateFormat: string;
12470:     LongDateFormat: string;
12471:     TimeAMString: string;
12472:     TimePMString: string;
12473:     ShortTimeFormat: string;
12474:     LongTimeFormat: string;
12475:     ShortMonthNames: array[1..12] of string;
12476:     LongMonthNames: array[1..12] of string;
12477:     ShortDayNames: array[1..7] of string;
12478:     LongDayNames: array[1..7] of string;
12479:     TwoDigitYearCenturyWindow: Word;
12480:   end;
12481:
12482: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12483: begin
12484:  Function ovFormatSettings : TFormatSettings
12485: end;
12486:
12487: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12488: begin
12489:   TOvcCharSet', 'set of Char
12490:   ovBTable', 'array[0..255] of Byte
12491:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
       HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12492:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12493:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12494:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12495:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12496:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12497:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12498:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12499:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12500:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12501:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12502:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12503:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12504:  Function LoCaseChar( C : Char) : Char
12505:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12506:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12507:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12508:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12509:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12510:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12511:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12512:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12513:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12514:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12515:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12516:  Procedure TrimAllSpacesPChar( P : PChar)
12517:  Function TrimEmbeddedZeros( const S : string) : string
12518:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12519:  Function TrimTrailPrimPChar( S : PChar) : PChar
12520:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12521:  Function TrimTrailingZeros( const S : string) : string
12522:  Procedure TrimTrailingZerosPChar( P : PChar)
12523:  Function UpCaseChar( C : Char) : Char
12524:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12525:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12526:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12527: end;
12528:
12529: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12530: begin
12531:   //PRaiseFrame', '^TRaiseFrame // will not work
12532:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12533:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12534:  Procedure SafeCloseHandle( var Handle : THandle)
12535:  Procedure ExchangeInteger( X1, X2 : Integer)
12536:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12537:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12538:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12539:
12540: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12541:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12542: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12543:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12544:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12545:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12546:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12547:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12548:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12549:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12550:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12551:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12552:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12553:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12554:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
```

```
12555:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12556:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12557:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12558:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12559:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12560:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12561:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12562:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12563:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12564:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12565:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12566:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12567:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12568:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12569:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12570:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12571:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12572:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12573:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12574:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12575:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12576:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12577:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12578:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12579:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12580:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12581:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12582:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12583:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12584:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12585:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12586:       var lpLuid: TLargeInteger): BOOL; stdcall;
12587:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12588:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12589:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12590:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12591:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12592:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12593:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12594:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12595:       var GenerateOnClose: BOOL): BOOL; stdcall;
12596:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12597:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12598:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12599:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12600:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12601:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12602:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12603:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12604:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12605:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12606:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12607:       var phkResult: HKEY): Longint; stdcall;
12608:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12609:       var phkResult: HKEY): Longint; stdcall;
12610:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12611:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12612:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12613:       lpdwDisposition: PDWORD): Longint; stdcall;
12614:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12615:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12616:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12617:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12618:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12619:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12620:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12621:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12622:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12623:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12624:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12625:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12626:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12627:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12628:       lpcbClass: PDWORD; lpReserved: Pointer;
12629:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12630:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12631:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12632:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12633:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12634:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12635:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12636:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12637:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12638:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12639:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12640:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12641:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12642:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12643:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
```

```
12644:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12645:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12646:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12647:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12648:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12649:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12650:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12651:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12652:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12653:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12654:
12655:  Function wAddAtom( lpString : PKOLChar) : ATOM
12656:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12657:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
      lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12658:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12659:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
      lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12660:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12661:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12662:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12663:  Function wCreateDirectoryEx(lpTemplateDirectory,
      lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12664:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
      bInitialState:BOOL;lpName:PKOLChar):THandle;
12665:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
      PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12666:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
      dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12667:  Function wCreateHardLink(lpFileName,
      lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12668:  Function
      CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12669:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
      nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12670:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
      lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
      Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
      lpProcessInfo:TProcessInformation):BOOL
12671:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
      Longint; lpName : PKOLChar) : THandle
12672:  Function
      wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12673:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12674:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12675:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12676:  //Function
      wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12677:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12678:  //Function
      wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12679:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12680:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12681:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12682:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12683:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12684:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12685:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
      dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12686:  Function wFindAtom( lpString : PKOLChar) : ATOM
12687:  Function
      wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12688:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12689:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
      Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12690:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12691:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12692:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12693:  Function
      wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12694:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
      DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12695:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12696:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12697:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12698:  Function wGetCommandLine : PKOLChar
12699:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12700:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12701:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12702:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
      PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12703:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12704:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
      lpDateStr : PKOLChar; cchDate : Integer) : Integer
12705:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12706:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
      lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12707:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
      lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
```

```
12708:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12709:  Function wGetEnvironmentStrings : PKOLChar
12710:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12711:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12712:  //Function
       wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12713:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
       lpFilePart:PKOLChar):DWORD;
12714:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12715:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12716:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12717:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12718:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
       lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12719:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
       lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12720:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12721:  Function
       wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12722:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12723:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
       nSize:DWORD; lpFileName : PKOLChar) : DWORD
12724:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12725:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12726:  Function wGetProfileString(lpAppName,lpKeyName,
       lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12727:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12728:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12729: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
       lpCharType):BOOL
12730:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12731:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12732:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12733: //Function
       wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12734: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12735: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
       : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
       lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12736:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12737:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12738:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12739:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12740:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12741:  Function
       wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12742:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12743:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12744:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12745:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12746:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
       TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12747:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12748:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12749:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12750:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12751:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12752:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12753:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
       lpNumberOfEventsRead:DWORD):BOOL;
12754:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12755:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12756:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
       lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12757:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
       lpNumbOfEventsRead:DWORD):BOOL;
12758:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
       : TCoord; var lpReadRegion : TSmallRect) : BOOL
12759:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
       DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12760:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12761:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
       lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12762:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
       lpFilePart:PKOLChar):DWORD;
12763:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12764:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12765:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12766:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12767:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12768:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12769:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12770:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12771:  //Function wUpdateResource(hUpdate:THandle;lpType,
       lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12772:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12773:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12774:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
       DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
```

```
12775:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
        var lpNumberOfEventsWritten : DWORD) : BOOL
12776:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
        TCoord; var lpWriteRegion : TSmallRect) : BOOL
12777:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12778:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12779:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12780:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12781:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12782:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12783:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12784:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12785:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12786:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12787:  Function wlstrlen( lpString : PKOLChar) : Integer
12788:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
        PNetConnectInfoStruct) : DWORD
12789:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
        lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12790:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12791:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12792:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12793:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12794:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12795:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12796:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12797:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12798:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
        : PKOLChar; nNameBufSize : DWORD) : DWORD
12799:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12800:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12801:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12802: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
        lpBufferSize:DWORD):DWORD;
12803:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12804: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12805: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12806:  //Function wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12807:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12808:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12809:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
        lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12810:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
        szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12811: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12812: //Function wGetPrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12813: //Function wWritePrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12814:  Function wAddFontResource( FileName : PKOLChar) : Integer
12815:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12816:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12817:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12818:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12819:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12820: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12821:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
        fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
        fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12822:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12823:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12824: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12825:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12826:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12827: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
        pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12828: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12829:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12830:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12831:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12832:  //Function wExtTextOut(DC:HDC;X,
        Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12833:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12834:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12835:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12836:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12837: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12838: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12839:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12840:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12841: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12842: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
        lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12843:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12844: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
```

```
12845:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12846: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12847:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12848:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12849:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12850:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12851:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12852:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12853:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12854:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12855:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12856:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12857:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12858:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12859:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12860:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12861:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12862:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12863:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12864:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12865:  //Function
       wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12866:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12867: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
       dwFlags : DWORD; lParam : Pointer) : Longint
12868:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12869:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12870:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12871:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12872:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12873:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12874: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12875:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12876:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12877:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12878:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12879:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12880:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12881:  //Function wCreateDesktop(lpszDesktop,
       lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12882:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
       HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12883:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
       lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12884:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
       hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12885: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
       nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12886:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
       dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12887:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12888:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12889:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12890:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12891:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
       : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12892:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
       : TFNDlgProc; dwInitParam : LPARAM) : Integer
12893:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12894:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
       nIDStaticPath:Integer;uFileType:UINT):Integer;
12895:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
       nIDStaticPath:Int;uFiletype:UINT):Int;
12896:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12897:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12898: //Function wDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
       cy:Int;Flags:UINT):BOOL;
12899:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12900:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12901:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12902:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
       pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12903: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12904:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12905:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12906:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12907:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12908:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12909:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12910:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12911:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12912:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12913:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12914:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12915:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
       lpnTabStopPositions) : DWORD
12916: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
       lpnLengthNeed:DWORD)BOOL;
12917:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
```

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

```
13003:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
13004:   TAfSafeDirectSyncEvent', 'Procedure
13005:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13006:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
13007:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
13008:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
13009:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
13010:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
13011:  Function AfIsSyncMethod : Boolean
13012:  Function AfSyncWnd : HWnd
13013:  Function AfSyncStatistics : TAfSyncStatistics
13014:  Procedure AfClearSyncStatistics
13015: end;
13016:
13017: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13018: begin
13019:  'fBinary','LongWord')( $00000001);
13020:  'fParity','LongWord'( $00000002);
13021:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13022:  'fOutxDsrFlow','LongWord')( $00000008);
13023:  'fDtrControl','LongWord')( $00000030);
13024:  'fDtrControlDisable','LongWord')( $00000000);
13025:  'fDtrControlEnable','LongWord')( $00000010);
13026:  'fDtrControlHandshake','LongWord')( $00000020);
13027:  'fDsrSensitivity','LongWord')( $00000040);
13028:  'fTXContinueOnXoff','LongWord')( $00000080);
13029:  'fOutX','LongWord')( $00000100);
13030:  'fInX','LongWord')( $00000200);
13031:  'fErrorChar','LongWord')( $00000400);
13032:  'fNull','LongWord')( $00000800);
13033:  'fRtsControl','LongWord')( $00003000);
13034:  'fRtsControlDisable','LongWord')( $00000000);
13035:  'fRtsControlEnable','LongWord')( $00001000);
13036:  'fRtsControlHandshake','LongWord')( $00002000);
13037:  'fRtsControlToggle','LongWord')( $00003000);
13038:  'fAbortOnError','LongWord')( $00004000);
13039:  'fDummy2','LongWord')( $FFFF8000);
13040:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13041:   FindClass('TOBJECT'),'EAfComPortCoreError
13042:   FindClass('TOBJECT'),'TAfComPortCore
13043:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
13044:    +'tKind : TAfCoreEvent; Data : DWORD)
13045:   SIRegister_TAfComPortCoreThread(CL);
13046:   SIRegister_TAfComPortEventThread(CL);
13047:   SIRegister_TAfComPortWriteThread(CL);
13048:   SIRegister_TAfComPortCore(CL);
13049:  Function FormatDeviceName( PortNumber : Integer) : string
13050: end;
13051:
13052: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13053: begin
13054:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13055:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13056:   SIRegister_TApplicationFileIO(CL);
13057:   TDataFileCapability', '( dfcRead, dfcWrite )
13058:   TDataFileCapabilities', 'set of TDataFileCapability
13059:   SIRegister_TDataFile(CL);
13060:   //TDataFileClass', 'class of TDataFile
13061:  Function ApplicationFileIODefined : Boolean
13062:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13063:  Function FileStreamExists(const fileName: String) : Boolean
13064:  //Procedure Register
13065: end;
13066:
13067: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13068: begin
13069:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13070:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13071:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13072:   TALFBXScale', 'Integer
13073:   FindClass('TOBJECT'),'EALFBXConvertError
13074:   SIRegister_EALFBXError(CL);
13075:   SIRegister_EALFBXException(CL);
13076:   FindClass('TOBJECT'),'EALFBXGFixError
13077:   FindClass('TOBJECT'),'EALFBXDSQLError
13078:   FindClass('TOBJECT'),'EALFBXDynError
13079:   FindClass('TOBJECT'),'EALFBXGBakError
13080:   FindClass('TOBJECT'),'EALFBXGSecError
13081:   FindClass('TOBJECT'),'EALFBXLicenseError
13082:   FindClass('TOBJECT'),'EALFBXGStatError
13083:   //EALFBXExceptionClass', 'class of EALFBXError
13084:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13085:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13086:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13087:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13088:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13089:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13090:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13091:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
```

```
13092:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13093:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13094:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13095:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13096:   TALFBXTransParams', 'set of TALFBXTransParam
13097:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13098:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13099:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13100: 'cALFBXMaxParamLength','LongInt'( 125);
13101:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13102:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13103:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13104:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13105:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13106:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13107:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13108:   SIRegister_TALFBXSQLDA(CL);
13109:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13110:   SIRegister_TALFBXPoolStream(CL);
13111:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13112:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13113:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13114:   //TALFBXArrayDesc', 'TISCArrayDesc
13115:   //TALFBXBlobDesc', 'TISCBlobDesc
13116:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13117:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13118:   SIRegister_TALFBXSQLResult(CL);
13119:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13120:   SIRegister_TALFBXSQLParams(CL);
13121:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13122:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13123:    +'atementType : TALFBXStatementType; end
13124:   FindClass('TOBJECT'),'TALFBXLibrary
13125:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13126:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13127:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13128:    //+' Excep : EALFBXExceptionClass)
13129:   SIRegister_TALFBXLibrary(CL);
13130:  'cAlFBXDateOffset','LongInt'( 15018);
13131:  'cALFBXTimeCoeff','LongInt'( 864000000);
13132:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13133:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13134:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13135:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13136:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13137:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13138:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13139:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13140:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13141:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13142:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13143:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13144:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13145:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13146: end;
13147:
13148: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13149: begin
13150:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13151:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13152:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13153:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13154:    +'teger; First : Integer; CacheThreshold : Integer; end
13155:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13156:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13157:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13158:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13159:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13160:   SIRegister_TALFBXClient(CL);
13161:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13162:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13163:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13164:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13165:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13166:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13167:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13168:   SIRegister_TALFBXConnectionPoolClient(CL);
13169:   SIRegister_TALFBXEventThread(CL);
13170:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13171: end;
13172:
13173: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13174: begin
13175: _OSVERSIONINFOA = record
13176:     dwOSVersionInfoSize: DWORD;
13177:     dwMajorVersion: DWORD;
13178:     dwMinorVersion: DWORD;
13179:     dwBuildNumber: DWORD;
13180:     dwPlatformId: DWORD;
```

```
13181:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13182:   end;
13183:  TOSVersionInfoA', '_OSVERSIONINFOA
13184:   TOSVersionInfo', 'TOSVersionInfoA
13185:  'WS_EX_RIGHT','LongWord')( $00001000);
13186:  'WS_EX_LEFT','LongWord')( $00000000);
13187:  'WS_EX_RTLREADING','LongWord')( $00002000);
13188:  'WS_EX_LTRREADING','LongWord')( $00000000);
13189:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13190:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13191:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13192:  'LAYOUT_RTL','LongWord')( $00000001);
13193:  'LAYOUT_BTT','LongWord')( $00000002);
13194:  'LAYOUT_VBH','LongWord')( $00000004);
13195:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13196:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13197:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13198:  Function GetLayout( dc : hdc) : DWORD
13199:  Function IsBidi : Boolean
13200:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13201:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13202:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13203:  Function GetPriorityClass( hProcess : THandle) : DWORD
13204:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13205:  Function CloseClipboard : BOOL
13206:  Function GetClipboardSequenceNumber : DWORD
13207:  Function GetClipboardOwner : HWND
13208:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13209:  Function GetClipboardViewer : HWND
13210:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13211:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13212:  Function GetClipboardData( uFormat : UINT) : THandle
13213:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13214:  Function CountClipboardFormats : Integer
13215:  Function EnumClipboardFormats( format : UINT) : UINT
13216:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13217:  Function EmptyClipboard : BOOL
13218:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13219:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13220:  Function GetOpenClipboardWindow : HWND
13221:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13222:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13223:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13224:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13225:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13226:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13227:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13228:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13229:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13230: end;
13231:
13232: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13233: begin
13234:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13235:  Function GetTemporaryFilesPath : String
13236:  Function GetTemporaryFileName : String
13237:  Function FindFileInPaths( const fileName, paths : String) : String
13238:  Function PathsToString( const paths : TStrings) : String
13239:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13240:  //Function MacroExpandPath( const aPath : String) : String
13241: end;
13242:
13243: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13244: begin
13245:   SIRegister_TALMultiPartBaseContent(CL);
13246:   SIRegister_TALMultiPartBaseContents(CL);
13247:   SIRegister_TAlMultiPartBaseStream(CL);
13248:   SIRegister_TALMultipartBaseEncoder(CL);
13249:   SIRegister_TALMultipartBaseDecoder(CL);
13250:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13251:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13252:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13253: end;
13254:
13255: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13256: begin
13257:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13258:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13259:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13260:  Function aAllocPadedMem( Size : Cardinal) : TObject
13261:  Procedure aFreePadedMem( var P : TObject);
13262:  Procedure aFreePadedMem1( var P : PChar);
13263:  Function aCheckPadedMem( P : Pointer) : Byte
13264:  Function aGetPadMemSize( P : Pointer) : Cardinal
13265:  Function aAllocMem( Size : Cardinal) : Pointer
13266:  Function aStrLen( const Str : PChar) : Cardinal
13267:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13268:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13269:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
```

```
13270:  Function aStrEnd( const Str : PChar) : PChar
13271:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13272:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13273:  Function aPCharLength( const Str : PChar) : Cardinal
13274:  Function aPCharUpper( Str : PChar) : PChar
13275:  Function aPCharLower( Str : PChar) : PChar
13276:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13277:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13278:  Function aCopyTail( const S : String; Len : Integer) : String
13279:  Function aInt2Thos( I : Int64) : String
13280:  Function aUpperCase( const S : String) : String
13281:  Function aLowerCase( const S : string) : String
13282:  Function aCompareText( const S1, S2 : string) : Integer
13283:  Function aSameText( const S1, S2 : string) : Boolean
13284:  Function aInt2Str( Value : Int64) : String
13285:  Function aStr2Int( const Value : String) : Int64
13286:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13287:  Function aGetFileExt( const FileName : String) : String
13288:  Function aGetFilePath( const FileName : String) : String
13289:  Function aGetFileName( const FileName : String) : String
13290:  Function aChangeExt( const FileName, Extension : String) : String
13291:  Function aAdjustLineBreaks( const S : string) : string
13292:  Function aGetWindowStr( WinHandle : HWND) : String
13293:  Function aDiskSpace( Drive : String) : TdriveSize
13294:  Function aFileExists( FileName : String) : Boolean
13295:  Function aFileSize( FileName : String) : Int64
13296:  Function aDirectoryExists( const Name : string) : Boolean
13297:  Function aSysErrorMessage( ErrorCode : Integer) : string
13298:  Function aShortPathName( const LongName : string) : string
13299:  Function aGetWindowVer : TWinVerRec
13300:  procedure InitDriveSpacePtr;
13301: end;
13302:
13303: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13304: begin
13305:  aZero','LongInt'( 0);
13306:  'makeappDEF','LongInt'( - 1);
13307:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13308:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13309:  'CS_KEYCVTWINDOW','LongInt'( 4);
13310:  'CS_DBLCLKS','LongInt'( 8);
13311:  'CS_OWNDC','LongWord')( $20);
13312:  'CS_CLASSDC','LongWord')( $40);
13313:  'CS_PARENTDC','LongWord')( $80);
13314:  'CS_NOKEYCVT','LongWord')( $100);
13315:  'CS_NOCLOSE','LongWord')( $200);
13316:  'CS_SAVEBITS','LongWord')( $800);
13317:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13318:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13319:  'CS_GLOBALCLASS','LongWord')( $4000);
13320:  'CS_IME','LongWord')( $10000);
13321:  'CS_DROPSHADOW','LongWord')( $20000);
13322:   //PPanelFunc', '^TPanelFunc // will not work
13323:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13324:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13325:   TFontLooks', 'set of TFontLook
13326:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13327:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13328:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13329:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13330:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13331:  Procedure RunMsgLoop( Show : Boolean)
13332:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13333:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
      ID_Number:Cardinal;hFont:Int):Int;
13334:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13335:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13336:  Function MakePanel(Left,Top,Width,Height,
      hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13337:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13338:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13339:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13340: end;
13341:
13342: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13343: begin
13344:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13345:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13346:   TScreenSaverOptions', 'set of TScreenSaverOption
13347:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
      ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13348:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13349:   SIRegister_TScreenSaver(CL);
13350:  //Procedure Register
13351:  Procedure SetScreenSaverPassword
13352: end;
13353:
13354: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13355: begin
```

```
13356:   FindClass('TOBJECT'),'TXCollection
13357:   SIRegister_EFilerException(CL);
13358:   SIRegister_TXCollectionItem(CL);
13359:   //TXCollectionItemClass', 'class of TXCollectionItem
13360:   SIRegister_TXCollection(CL);
13361:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13362:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13363:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13364:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13365:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13366:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13367: end;
13368:
13369: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13370: begin
13371:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13372:  Procedure xglMapTexCoordToNull
13373:  Procedure xglMapTexCoordToMain
13374:  Procedure xglMapTexCoordToSecond
13375:  Procedure xglMapTexCoordToDual
13376:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13377:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13378:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13379:  Procedure xglBeginUpdate
13380:  Procedure xglEndUpdate
13381:  Procedure xglPushState
13382:  Procedure xglPopState
13383:  Procedure xglForbidSecondTextureUnit
13384:  Procedure xglAllowSecondTextureUnit
13385:  Function xglGetBitWiseMapping : Cardinal
13386: end;
13387:
13388: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13389: begin
13390:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13391:   TBaseListOptions', 'set of TBaseListOption
13392:   SIRegister_TBaseList(CL);
13393:   SIRegister_TBaseVectorList(CL);
13394:   SIRegister_TAffineVectorList(CL);
13395:   SIRegister_TVectorList(CL);
13396:   SIRegister_TTexPointList(CL);
13397:   SIRegister_TXIntegerList(CL);
13398:   //PSingleArrayList', '^TSingleArrayList // will not work
13399:   SIRegister_TSingleList(CL);
13400:   SIRegister_TByteList(CL);
13401:   SIRegister_TQuaternionList(CL);
13402:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13403:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13404:  Procedure FastQuickSortLists(startIndex,
      endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13405: end;
13406:
13407: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13408: begin
13409:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13410:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13411:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
      indices:TIntegerList;list:TAffineVectorList);
13412:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
      indices:TIntegerList;list:TAffineVectorList);
13413:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
      normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13414:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13415:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13416:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13417:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13418:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13419:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13420:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13421:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13422:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13423:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
      edgesTriangles : TIntegerList) : TIntegerList
13424:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13425:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
      TPersistentObjectList;
13426:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13427:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
      TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13428: end;
13429:
13430: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13431: begin
13432:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13433:  Procedure FreeMemAndNil( var P : TObject)
13434:  Function PCharOrNil( const S : string) : PChar
13435:   SIRegister_TJclReferenceMemoryStream(CL);
13436:   FindClass('TOBJECT'),'EJclVMTError
13437:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
```

```
13438:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13439:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13440:   PDynamicIndexList', '^TDynamicIndexList // will not work
13441:   PDynamicAddressList', '^TDynamicAddressList // will not work
13442:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13443:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13444:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13445:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13446:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13447:  Function GetInitTable( AClass : TClass) : PTypeInfo
13448:   PFieldEntry', '^TFieldEntry // will not work}
13449:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13450:  Function JIsClass( Address : Pointer) : Boolean
13451:  Function JIsObject( Address : Pointer) : Boolean
13452:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13453:   TDigitCount', 'Integer
13454:   SIRegister_TJclNumericFormat(CL);
13455:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13456:   TTextHandler', 'Procedure ( const Text : string)
13457: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13458:  Function JExecute(const
       CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13459:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13460:  Function ReadKey : Char   //to and from the DOS console !
13461:   TModuleHandle', 'HINST
13462:   //TModuleHandle', 'Pointer
13463:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13464:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13465:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13466:  Procedure UnloadModule( var Module : TModuleHandle)
13467:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13468:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13469:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13470:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13471:   FindClass('TOBJECT'),'EJclConversionError
13472:  Function JStrToBoolean( const S : string) : Boolean
13473:  Function JBooleanToStr( B : Boolean) : string
13474:  Function JIntToBool( I : Integer) : Boolean
13475:  Function JBoolToInt( B : Boolean) : Integer
13476:  'ListSeparator','String ';
13477:  'ListSeparator1','String ':
13478:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13479:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13480:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13481:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13482:  Function ListItemCount( const List, Separator : string) : Integer
13483:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13484:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13485:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13486:  Function SystemTObjectInstance : LongWord
13487:  Function IsCompiledWithPackages : Boolean
13488:  Function JJclGUIDToString( const GUID : TGUID) : string
13489:  Function JJclStringToGUID( const S : string) : TGUID
13490:   SIRegister_TJclIntfCriticalSection(CL);
13491:   SIRegister_TJclSimpleLog(CL);
13492:  Procedure InitSimpleLog( const ALogFileName : string)
13493: end;
13494:
13495: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13496: begin
13497:   FindClass('TOBJECT'),'EJclBorRADException
13498:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13499:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13500:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13501:   TJclBorRADToolPath', 'string
13502:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13503:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13504:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13505:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13506:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13507:  BorRADToolRepositoryFormsPage','String 'Forms
13508:  BorRADToolRepositoryProjectsPage','String 'Projects
13509:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13510:  BorRADToolRepositoryObjectType','String 'Type
13511:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13512:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13513:  BorRADToolRepositoryObjectName','String 'Name
13514:  BorRADToolRepositoryObjectPage','String 'Page
13515:  BorRADToolRepositoryObjectIcon','String 'Icon
13516:  BorRADToolRepositoryObjectDescr','String 'Description
13517:  BorRADToolRepositoryObjectAuthor','String 'Author
13518:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13519:  BorRADToolRepositoryObjectDesigner','String 'Designer
13520:  BorRADToolRepositoryDesignerDfm','String 'dfm
13521:  BorRADToolRepositoryDesignerXfm','String 'xfm
13522:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13523:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13524:  SourceExtensionDelphiPackage','String '.dpk
13525:  SourceExtensionBCBPackage','String '.bpk
```

```
13526:  SourceExtensionDelphiProject','String '.dpr
13527:  SourceExtensionBCBProject','String '.bpr
13528:  SourceExtensionBDSProject','String '.bdsproj
13529:  SourceExtensionDProject','String '.dproj
13530:  BinaryExtensionPackage','String '.bpl
13531:  BinaryExtensionLibrary','String '.dll
13532:  BinaryExtensionExecutable','String '.exe
13533:  CompilerExtensionDCP','String '.dcp
13534:  CompilerExtensionBPI','String '.bpi
13535:  CompilerExtensionLIB','String '.lib
13536:  CompilerExtensionTDS','String '.tds
13537:  CompilerExtensionMAP','String '.map
13538:  CompilerExtensionDRC','String '.drc
13539:  CompilerExtensionDEF','String '.def
13540:  SourceExtensionCPP','String '.cpp
13541:  SourceExtensionH','String '.h
13542:  SourceExtensionPAS','String '.pas
13543:  SourceExtensionDFM','String '.dfm
13544:  SourceExtensionXFM','String '.xfm
13545:  SourceDescriptionPAS','String 'Pascal source file
13546:  SourceDescriptionCPP','String 'C++ source file
13547:  DesignerVCL','String 'VCL
13548:  DesignerCLX','String 'CLX
13549:  ProjectTypePackage','String 'package
13550:  ProjectTypeLibrary','String 'library
13551:  ProjectTypeProgram','String 'program
13552:  Personality32Bit','String '32 bit
13553:  Personality64Bit','String '64 bit
13554:  PersonalityDelphi','String 'Delphi
13555:  PersonalityDelphiDotNet','String 'Delphi.net
13556:  PersonalityBCB','String 'C++Builder
13557:  PersonalityCSB','String 'C#Builder
13558:  PersonalityVB','String 'Visual Basic
13559:  PersonalityDesign','String 'Design
13560:  PersonalityUnknown','String 'Unknown personality
13561:  PersonalityBDS','String 'Borland Developer Studio
13562:  DOFDirectoriesSection','String 'Directories
13563:  DOFUnitOutputDirKey','String 'UnitOutputDir
13564:  DOFSearchPathName','String 'SearchPath
13565:  DOFConditionals','String 'Conditionals
13566:  DOFLinkerSection','String 'Linker
13567:  DOFPackagesKey','String 'Packages
13568:  DOFCompilerSection','String 'Compiler
13569:  DOFPackageNoLinkKey','String 'PackageNoLink
13570:  DOFAdditionalSection','String 'Additional
13571:  DOFOptionsKey','String 'Options
13572:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13573:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13574:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13575:   TJclBorPersonalities', 'set of TJclBorPersonality
13576:   TJclBorDesigner', '( bdVCL, bdCLX )
13577:   TJclBorDesigners', 'set of TJClBorDesigner
13578:   TJclBorPlatform', '( bp32bit, bp64bit )
13579:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13580:   SIRegister_TJclBorRADToolInstallationObject(CL);
13581:   SIRegister_TJclBorlandOpenHelp(CL);
13582:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13583:   TJclHelp2Objects', 'set of TJclHelp2Object
13584:   SIRegister_TJclHelp2Manager(CL);
13585:   SIRegister_TJclBorRADToolIdeTool(CL);
13586:   SIRegister_TJclBorRADToolIdePackages(CL);
13587:   SIRegister_IJclCommandLineTool(CL);
13588:   FindClass('TOBJECT'),'EJclCommandLineToolError
13589:   SIRegister_TJclCommandLineTool(CL);
13590:   SIRegister_TJclBorlandCommandLineTool(CL);
13591:   SIRegister_TJclBCC32(CL);
13592:   SIRegister_TJclDCC32(CL);
13593:   TJclDCC', 'TJclDCC32
13594:   SIRegister_TJclBpr2Mak(CL);
13595:   SIRegister_TJclBorlandMake(CL);
13596:   SIRegister_TJclBorRADToolPalette(CL);
13597:   SIRegister_TJclBorRADToolRepository(CL);
13598:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13599:   TCommandLineTools', 'set of TCommandLineTool
13600:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13601:   SIRegister_TJclBorRADToolInstallation(CL);
13602:   SIRegister_TJclBCBInstallation(CL);
13603:   SIRegister_TJclDelphiInstallation(CL);
13604:   SIRegister_TJclDCCIL(CL);
13605:   SIRegister_TJclBDSInstallation(CL);
13606:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13607:   SIRegister_TJclBorRADToolInstallations(CL);
13608:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13609:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13610:  Function IsDelphiPackage( const FileName : string) : Boolean
13611:  Function IsDelphiProject( const FileName : string) : Boolean
13612:  Function IsBCBPackage( const FileName : string) : Boolean
13613:  Function IsBCBProject( const FileName : string) : Boolean
13614:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
```

```
13615:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13616:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
       Descript:PString;
13617:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString
13618:  function SamePath(const Path1, Path2: string): Boolean;
13619: end;
13620:
13621: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13622: begin
13623:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13624:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13625:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13626:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13627:  'LPathSeparator','String '/
13628:  'LDirDelimiter','String '/
13629:  'LDirSeparator','String ':
13630:  'JXPathDevicePrefix','String '\\.\
13631:  'JXPathSeparator','String '\
13632:  'JXDirDelimiter','String '\
13633:  'JXDirSeparator','String ';
13634:  'JXPathUncPrefix','String '\\
13635:  'faNormalFile','LongWord')( $00000080);
13636:  //'faUnixSpecific',' faSymLink);
13637:   JXTCompactPath', '( cpCenter, cpEnd )
13638:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13639:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13640:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13641:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13642:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13643:
13644:  Function jxPathAddSeparator( const Path : string) : string
13645:  Function jxPathAddExtension( const Path, Extension : string) : string
13646:  Function jxPathAppend( const Path, Append : string) : string
13647:  Function jxPathBuildRoot( const Drive : Byte) : string
13648:  Function jxPathCanonicalize( const Path : string) : string
13649:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13650:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13651:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13652:  Function jxPathExtractFileDirFixed( const S : string) : string
13653:  Function jxPathExtractFileNameNoExt( const Path : string) : string
13654:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13655:  Function jxPathGetDepth( const Path : string) : Integer
13656:  Function jxPathGetLongName( const Path : string) : string
13657:  Function jxPathGetShortName( const Path : string) : string
13658:  Function jxPathGetLongName( const Path : string) : string
13659:  Function jxPathGetShortName( const Path : string) : string
13660:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13661:  Function jxPathGetTempPath : string
13662:  Function jxPathIsAbsolute( const Path : string) : Boolean
13663:  Function jxPathIsChild( const Path, Base : string) : Boolean
13664:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13665:  Function jxPathIsUNC( const Path : string) : Boolean
13666:  Function jxPathRemoveSeparator( const Path : string) : string
13667:  Function jxPathRemoveExtension( const Path : string) : string
13668:  Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13669:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13670:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13671:   JxTFileListOptions', 'set of TFileListOption
13672:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13673:   TFileHandler', 'Procedure ( const FileName : string)
13674:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13675:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13676:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
       AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
       FileMatchFunc:TFileMatchFunc):Bool;
13677:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13678:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13679:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13680:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13681:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
       RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13682:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
       IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13683:  Procedure jxCreateEmptyFile( const FileName : string)
13684:  Function jxCloseVolume( var Volume : THandle) : Boolean
13685:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13686:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13687:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13688:  Function jxDelTree( const Path : string) : Boolean
13689:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13690:  Function jxDiskInDrive( Drive : Char) : Boolean
13691:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13692:  Function jxFileCreateTemp( var Prefix : string) : THandle
13693:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13694:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13695:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13696:  Function jxFileExists( const FileName : string) : Boolean
13697:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
```

```
13698:  Function jxFileRestore( const FileName : string) : Boolean
13699:  Function jxGetBackupFileName( const FileName : string) : string
13700:  Function jxIsBackupFileName( const FileName : string) : Boolean
13701:  Function jxFileGetDisplayName( const FileName : string) : string
13702:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13703:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13704:  Function jxFileGetSize( const FileName : string) : Int64
13705:  Function jxFileGetTempName( const Prefix : string) : string
13706:  Function jxFileGetTypeName( const FileName : string) : string
13707:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13708:  Function jxForceDirectories( Name : string) : Boolean
13709:  Function jxGetDirectorySize( const Path : string) : Int64
13710:  Function jxGetDriveTypeStr( const Drive : Char) : string
13711:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13712:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13713:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13714:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13715:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13716:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
       ResolveSymLinks:Boolean):Integer
13717:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13718:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13719:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13720:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13721:  Function jxGetFileCreation( const FName : string) : TFileTime;
13722:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13723:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13724:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13725:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13726:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13727:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13728:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13729:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13730:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13731:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13732:  Function jxGetModulePath( const Module : HMODULE) : string
13733:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13734:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13735:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13736:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13737:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13738:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13739:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13740:  Function jxOpenVolume( const Drive : Char) : THandle
13741:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13742:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13743:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13744:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13745:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13746:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13747:  Procedure jxShredFile( const FileName : string; Times : Integer)
13748:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13749:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13750:  Function jxSymbolicLinkTarget( const Name : string) : string
13751:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13752:   SIRegister_TJclCustomFileAttrMask(CL);
13753:   SIRegister_TJclFileAttributeMask(CL);
13754:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13755:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13756:   TFileSearchOptions', 'set of TFileSearchOption
13757:   TFileSearchTaskID', 'Integer
13758:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13759:    +'hTaskID; const Aborted : Boolean)
13760:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13761:   SIRegister_IJclFileEnumerator(CL);
13762:   SIRegister_TJclFileEnumerator(CL);
13763:  Function JxFileSearch : IJclFileEnumerator
13764:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13765:   JxTFileFlags', 'set of TFileFlag
13766:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13767:   SIRegister_TJclFileVersionInfo(CL);
13768:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13769:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13770:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13771:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13772:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13773:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13774:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13775:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13776:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
       Revision:Word);
13777:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13778:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
       NotAvailableText : string) : string
13779:   SIRegister_TJclTempFileStream(CL);
13780:   FindClass('TOBJECT'),'TJclCustomFileMapping
13781:   SIRegister_TJclFileMappingView(CL);
13782:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13783:   SIRegister_TJclCustomFileMapping(CL);
```

```
13784:   SIRegister_TJclFileMapping(CL);
13785:   SIRegister_TJclSwapFileMapping(CL);
13786:   SIRegister_TJclFileMappingStream(CL);
13787:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13788:   //PPCharArray', '^TPCharArray // will not work
13789:   SIRegister_TJclMappedTextReader(CL);
13790:   SIRegister_TJclFileMaskComparator(CL);
13791:   FindClass('TOBJECT'),'EJclPathError
13792:   FindClass('TOBJECT'),'EJclFileUtilsError
13793:   FindClass('TOBJECT'),'EJclTempFileStreamError
13794:   FindClass('TOBJECT'),'EJclTempFileStreamError
13795:   FindClass('TOBJECT'),'EJclFileMappingError
13796:   FindClass('TOBJECT'),'EJclFileMappingViewError
13797:  Function jxPathGetLongName2( const Path : string) : string
13798:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13799:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13800:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13801:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13802:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13803:  Procedure jxPathListAddItems( var List : string; const Items : string)
13804:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13805:  Procedure jxPathListDelItems( var List : string; const Items : string)
13806:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13807:  Function jxPathListItemCount( const List : string) : Integer
13808:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13809:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13810:  Function jxPathListItemIndex( const List, Item : string) : Integer
13811:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13812:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13813:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
       AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13814:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
       AllowedPrefixCharacters : string) : Integer
13815: end;
13816:
13817: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13818: begin
13819:  'UTF8FileHeader','String #$ef#$bb#$bf);
13820:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13821:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13822:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13823:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13824:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13825:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13826:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13827:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13828:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13829:  Function lFileIsReadable( const AFilename : string) : boolean
13830:  Function lFileIsWritable( const AFilename : string) : boolean
13831:  Function lFileIsText( const AFilename : string) : boolean
13832:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13833:  Function lFileIsExecutable( const AFilename : string) : boolean
13834:  Function lFileIsSymlink( const AFilename : string) : boolean
13835:  Function lFileIsHardLink( const AFilename : string) : boolean
13836:  Function lFileSize( const Filename : string) : int64;
13837:  Function lGetFileDescription( const AFilename : string) : string
13838:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13839:  Function lTryReadAllLinks( const Filename : string) : string
13840:  Function lDirPathExists( const FileName : String) : Boolean
13841:  Function lForceDirectory( DirectoryName : string) : boolean
13842:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13843:  Function lProgramDirectory : string
13844:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13845:  Function lExtractFileNameOnly( const AFilename : string) : string
13846:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13847:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13848:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13849:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13850:  Function lAppendPathDelim( const Path : string) : string
13851:  Function lChompPathDelim( const Path : string) : string
13852:  Function lTrimFilename( const AFilename : string) : string
13853:  Function lCleanAndExpandFilename( const Filename : string) : string
13854:  Function lCleanAndExpandDirectory( const Filename : string) : string
13855:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13856:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
       AlwaysRequireSharedBaseFolder : Boolean) : string
13857:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13858:  Function lFileIsInPath( const Filename, Path : string) : boolean
13859:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13860:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13861:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13862:  'AllDirectoryEntriesMask','String '*
13863:  Function lGetAllFilesMask : string
13864:  Function lGetExeExt : string
13865:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
       TSearchFileInPathFlags) : string
13866:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
       TSearchFileInPathFlags) : TStrings
13867:  Function lFindDiskFilename( const Filename : string) : string
```

```
13868:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13869:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13870:  Function lGetDarwinSystemFilename( Filename : string) : string
13871:   SIRegister_TFileIterator(CL);
13872:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13873:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13874:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13875:   SIRegister_TFileSearcher(CL);
13876:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13877:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13878:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13879:  // TCopyFileFlags', 'set of TCopyFileFlag
13880:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13881:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13882:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13883:  Function lReadFileToString( const Filename : string) : string
13884:  Function lGetTempFilename( const Directory, Prefix : string) : string
13885:  {Function NeedRTLAnsi : boolean
13886:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13887:  Function UTF8ToSys( const s : string) : string
13888:  Function SysToUTF8( const s : string) : string
13889:  Function ConsoleToUTF8( const s : string) : string
13890:  Function UTF8ToConsole( const s : string) : string}
13891:  Function FileExistsUTF8( const Filename : string) : boolean
13892:  Function FileAgeUTF8( const FileName : string) : Longint
13893:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13894:  Function ExpandFileNameUTF8( const FileName : string) : string
13895:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13896:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13897:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13898:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13899:  Procedure FindCloseUTF8( var F : TSearchrec)
13900:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13901:  Function FileGetAttrUTF8( const FileName : String) : Longint
13902:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13903:  Function DeleteFileUTF8( const FileName : String) : Boolean
13904:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13905:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13906:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13907:  Function GetCurrentDirUTF8 : String
13908:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13909:  Function CreateDirUTF8( const NewDir : String) : Boolean
13910:  Function RemoveDirUTF8( const Dir : String) : Boolean
13911:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13912:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13913:  Function FileCreateUTF8( const FileName : string) : THandle;
13914:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13915:  Function ParamStrUTF8( Param : Integer) : string
13916:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13917:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13918:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13919:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13920:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13921: end;
13922:
13923: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13924: begin
13925:   //VK_F23 = 134;
13926:   //{$EXTERNALSYM VK_F24}
13927:   //VK_F24 = 135;
13928:  TVirtualKeyCode', 'Integer
13929:  'VK_MOUSEWHEELUP','integer'(134);
13930:  'VK_MOUSEWHEELDOWN','integer'(135);
13931:  Function glIsKeyDown( c : Char) : Boolean;
13932:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13933:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13934:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13935:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13936:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13937:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13938: end;
13939:
13940: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13941: begin
13942:   TGLPoint', 'TPoint
13943:   //PGLPoint', '^TGLPoint // will not work
13944:   TGLRect', 'TRect
13945:   //PGLRect', '^TGLRect // will not work
13946:   TDelphiColor', 'TColor
13947:   TGLPicture', 'TPicture
13948:   TGLGraphic', 'TGraphic
13949:   TGLBitmap', 'TBitmap
13950:   //TGraphicClass', 'class of TGraphic
13951:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
13952:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
13953:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
13954:    +'Button; Shift : TShiftState; X, Y : Integer)
13955:   TGLMouseMoveEvent', 'TMouseMoveEvent
13956:   TGLKeyEvent', 'TKeyEvent
```

```
13957:   TGLKeyPressEvent', 'TKeyPressEvent
13958:   EGLOSError', 'EWin32Error
13959:   EGLOSError', 'EWin32Error
13960:   EGLOSError', 'EOSError
13961:  'glsAllFilter','string'All // sAllFilter
13962:  Function GLPoint( const x, y : Integer) : TGLPoint
13963:  Function GLRGB( const r, g, b : Byte) : TColor
13964:  Function GLColorToRGB( color : TColor) : TColor
13965:  Function GLGetRValue( rgb : DWORD) : Byte
13966:  Function GLGetGValue( rgb : DWORD) : Byte
13967:  Function GLGetBValue( rgb : DWORD) : Byte
13968:  Procedure GLInitWinColors
13969:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
13970:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
13971:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
13972:  Procedure GLInformationDlg( const msg : String)
13973:  Function GLQuestionDlg( const msg : String) : Boolean
13974:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
13975:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
13976:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
13977:  Function GLApplicationTerminated : Boolean
13978:  Procedure GLRaiseLastOSError
13979:  Procedure GLFreeAndNil( var anObject: TObject)
13980:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
13981:  Function GLGetCurrentColorDepth : Integer
13982:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
13983:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
13984:  Procedure GLSleep( length : Cardinal)
13985:  Procedure GLQueryPerformanceCounter( var val : Int64)
13986:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
13987:  Function GLStartPrecisionTimer : Int64
13988:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
13989:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
13990:  Function GLRDTSC : Int64
13991:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
13992:  Function GLOKMessageBox( const Text, Caption : string) : Integer
13993:  Procedure GLShowHTMLUrl( Url : String)
13994:  Procedure GLShowCursor( AShow : boolean)
13995:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
13996:  Procedure GLGetCursorPos( var point : TGLPoint)
13997:  Function GLGetScreenWidth : integer
13998:  Function GLGetScreenHeight : integer
13999:  Function GLGetTickCount : int64
14000:  function RemoveSpaces(const str : String) : String;
14001:   TNormalMapSpace','( nmsObject, nmsTangent )
14002:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14003:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14004:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
14005:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
14006:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14007: end;
14008:
14009: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14010: begin
14011:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14012:  // PGLStarRecord', '^TGLStarRecord // will not work
14013:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14014:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14015:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14016: end;
14017:
14018:
14019: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14020: begin
14021:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14022:   //PAABB', '^TAABB // will not work
14023:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14024:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14025:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14026:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14027:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14028:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14029:  Procedure SetAABB( var bb : TAABB; const v : TVector)
14030:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14031:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14032:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14033:  Function BBMinX( const c : THmgBoundingBox) : Single
14034:  Function BBMaxX( const c : THmgBoundingBox) : Single
14035:  Function BBMinY( const c : THmgBoundingBox) : Single
14036:  Function BBMaxY( const c : THmgBoundingBox) : Single
14037:  Function BBMinZ( const c : THmgBoundingBox) : Single
14038:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14039:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14040:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14041:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14042:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
```

```
14043:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14044:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14045:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14046:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14047:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14048:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14049:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14050:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14051:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14052:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14053:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14054:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14055:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14056:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14057:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14058:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14059:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14060:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14061:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14062:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14063:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14064:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14065:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
       testBSphere:TBSphere):TSpaceContains
14066:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14067:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14068:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14069:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14070:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14071:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
       viewportSizeY:Int):TClipRect
14072: end;
14073:
14074: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14075: begin
14076:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14077:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14078:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14079:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14080:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14081:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14082:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14083:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14084:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14085:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14086:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14087:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14088:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14089:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14090:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14091:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14092:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14093:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14094:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14095:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14096:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14097:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14098:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14099:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14100:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14101: end;
14102:
14103: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14104: begin
14105:  'EPSILON','Single').setExtended( 1e-40);
14106:  'EPSILON2','Single').setExtended( 1e-30);  }
14107:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14108:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14109:  THmgPlane', 'TVector
14110:   TDoubleHmgPlane', 'THomogeneousDblVector
14111:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14112:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14113:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14114:   TSingleArray', 'array of Single
14115:   TTransformations','array [0..15] of Single)
14116:   TPackedRotationMatrix','array [0..2] of Smallint)
14117:   TVertex', 'TAffineVector
14118:   //TVectorGL', 'THomogeneousFltVector
14119:   //TMatrixGL', 'THomogeneousFltMatrix
14120:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14121:  Function glTexPointMake( const s, t : Single) : TTexPoint
14122:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14123:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14124:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14125:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14126:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14127:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14128:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14129:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
```

```
14130:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14131:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14132:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14133:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14134:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14135:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14136:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14137:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14138:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14139:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14140:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14141:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14142:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14143:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14144:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14145:  Procedure glRstVector( var v : TAffineVector);
14146:  Procedure glRstVector1( var v : TVectorGL);
14147:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14148:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14149:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14150:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14151:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14152:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14153:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14154:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14155:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14156:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14157:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14158:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14159:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
       nb:Int;dest:PTexPointArray);
14160:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
       nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14161:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
       PAffineVectorArray);
14162:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14163:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14164:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14165:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14166:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14167:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14168:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14169:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14170:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14171:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14172:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14173:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14174:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14175:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14176:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14177:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14178:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14179:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14180:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14181:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14182:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14183:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14184:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14185:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14186:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14187:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14188:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14189:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14190:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14191:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14192:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14193:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14194:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14195:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14196:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14197:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14198:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14199:  Function glLerp( const start, stop, t : Single) : Single
14200:  Function glAngleLerp( start, stop, t : Single) : Single
14201:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14202:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14203:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14204:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14205:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14206:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14207:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14208:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14209: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14210: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
       PAffineVectorArray);
14211:  Function glVectorLength( const x, y : Single) : Single;
14212:  Function glVectorLength1( const x, y, z : Single) : Single;
14213:  Function glVectorLength2( const v : TAffineVector) : Single;
14214:  Function glVectorLength3( const v : TVectorGL) : Single;
```

```
14215:  Function glVectorLength4( const v : array of Single) : Single;
14216:  Function glVectorNorm( const x, y : Single) : Single;
14217:  Function glVectorNorm1( const v : TAffineVector) : Single;
14218:  Function glVectorNorm2( const v : TVectorGL) : Single;
14219:  Function glVectorNorm3( var V : array of Single) : Single;
14220:  Procedure glNormalizeVector( var v : TAffineVector);
14221:  Procedure glNormalizeVector1( var v : TVectorGL);
14222:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14223:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14224: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14225:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14226:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14227:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14228:  Procedure glNegateVector( var V : TAffineVector);
14229:  Procedure glNegateVector2( var V : TVectorGL);
14230:  Procedure glNegateVector3( var V : array of Single);
14231:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14232:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14233:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14234:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14235:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14236:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14237:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14238:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14239:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14240:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14241:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14242:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14243:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14244:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14245:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14246:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14247:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14248:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14249:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14250:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14251:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14252:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14253:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14254:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14255:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14256:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14257:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14258:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14259:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14260:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14261:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14262:  Procedure glAbsVector( var v : TVectorGL);
14263:  Procedure glAbsVector1( var v : TAffineVector);
14264:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14265:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14266:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14267:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14268:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14269:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14270:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14271:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14272:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14273:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14274:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14275:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14276:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14277:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14278:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14279:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14280:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14281:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14282:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14283:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14284:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14285:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14286:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14287:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14288:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14289:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14290:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14291:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14292:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14293:  Procedure glAdjointMatrix( var M : TMatrixGL);
14294:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14295:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14296:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14297:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14298:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14299:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14300:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14301:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14302:  Procedure glInvertMatrix( var M : TMatrixGL);
14303:  Procedure glInvertMatrix1( var M : TAffineMatrix);
```

```
14304:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14305:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14306:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14307:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14308:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14309:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14310:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14311:  Procedure glNormalizePlane( var plane : THmgPlane)
14312:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14313:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14314:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14315:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14316:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14317:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14318:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14319:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14320:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14321:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14322:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14323:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14324:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14325:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
       Segment0Closest, Segment1Closest : TAffineVector)
14326:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14327:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14328:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14329:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14330:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14331:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14332:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14333:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14334:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14335:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14336:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14337:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14338:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14339:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14340:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14341:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14342:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14343:  Function glLnXP1( X : Extended) : Extended
14344:  Function glLog10( X : Extended) : Extended
14345:  Function glLog2( X : Extended) : Extended;
14346:  Function glLog21( X : Single) : Single;
14347:  Function glLogN( Base, X : Extended) : Extended
14348:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14349:  Function glPower( const Base, Exponent : Single) : Single;
14350:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14351:  Function glDegToRad( const Degrees : Extended) : Extended;
14352:  Function glDegToRad1( const Degrees : Single) : Single;
14353:  Function glRadToDeg( const Radians : Extended) : Extended;
14354:  Function glRadToDeg1( const Radians : Single) : Single;
14355:  Function glNormalizeAngle( angle : Single) : Single
14356:  Function glNormalizeDegAngle( angle : Single) : Single
14357:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14358:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14359:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14360:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14361:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14362:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14363:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14364:  Function glArcCos( const X : Extended) : Extended;
14365:  Function glArcCos1( const x : Single) : Single;
14366:  Function glArcSin( const X : Extended) : Extended;
14367:  Function glArcSin1( const X : Single) : Single;
14368:  Function glArcTan21( const Y, X : Extended) : Extended;
14369:  Function glArcTan21( const Y, X : Single) : Single;
14370:  Function glFastArcTan2( y, x : Single) : Single
14371:  Function glTan( const X : Extended) : Extended;
14372:  Function glTan1( const X : Single) : Single;
14373:  Function glCoTan( const X : Extended) : Extended;
14374:  Function glCoTan1( const X : Single) : Single;
14375:  Function glSinh( const x : Single) : Single;
14376:  Function glSinh1( const x : Double) : Double;
14377:  Function glCosh( const x : Single) : Single;
14378:  Function glCosh1( const x : Double) : Double;
14379:  Function glRSqrt( v : Single) : Single
14380:  Function glRLength( x, y : Single) : Single
14381:  Function glISqrt( i : Integer) : Integer
14382:  Function glILength( x, y : Integer) : Integer;
14383:  Function glILength1( x, y, z : Integer) : Integer;
14384:  Procedure glRegisterBasedExp
14385:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14386:  Function glRoundInt( v : Single) : Single;
14387:  Function glRoundInt1( v : Extended) : Extended;
14388:  Function glTrunc( v : Single) : Integer;
14389:  Function glTrunc64( v : Extended) : Int64;
14390:  Function glInt( v : Single) : Single;
14391:  Function glInt1( v : Extended) : Extended;
```

```
14392:  Function glFrac( v : Single) : Single;
14393:  Function glFrac1( v : Extended) : Extended;
14394:  Function glRound( v : Single) : Integer;
14395:  Function glRound64( v : Single) : Int64;
14396:  Function glRound641( v : Extended) : Int64;
14397:  Function glTrunc( X : Extended) : Int64
14398:  Function glRound( X : Extended) : Int64
14399:  Function glFrac( X : Extended) : Extended
14400:  Function glCeil( v : Single) : Integer;
14401:  Function glCeil64( v : Extended) : Int64;
14402:  Function glFloor( v : Single) : Integer;
14403:  Function glFloor64( v : Extended) : Int64;
14404:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14405:  Function glSign( x : Single) : Integer
14406:  Function glIsInRange( const x, a, b : Single) : Boolean;
14407:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14408:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14409:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14410:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14411:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14412:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14413:  Function glMinFloat3( const v1, v2 : Single) : Single;
14414:  Function glMinFloat4( const v : array of Single) : Single;
14415:  Function glMinFloat5( const v1, v2 : Double) : Double;
14416:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14417:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14418:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14419:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14420:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14421:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14422:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14423:  Function glMaxFloat2( const v : array of Single) : Single;
14424:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14425:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14426:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14427:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14428:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14429:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14430:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14431:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14432:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14433:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14434:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14435:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14436:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14437:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14438:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14439:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14440:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14441:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14442:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14443:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14444:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14445:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14446:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14447:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14448:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14449:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14450:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14451:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14452:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14453:  Procedure glSortArrayAscending( var a : array of Extended)
14454:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14455:  Function glClampValue1( const aValue, aMin : Single) : Single;
14456:  Function glGeometryOptimizationMode : String
14457:  Procedure glBeginFPUOnlySection
14458:  Procedure glEndFPUOnlySection
14459:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14460:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14461:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14462:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14463:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14464:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14465:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14466:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14467:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14468:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14469:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14470:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14471:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14472:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14473:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14474:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14475:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14476:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14477:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14478:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
```

```
14479:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14480:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14481:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14482:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14483:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14484:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14485:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14486:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14487:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14488:  'cPI','Single').setExtended( 3.141592654);
14489:  'cPIdiv180','Single').setExtended( 0.017453292);
14490:  'c180divPI','Single').setExtended( 57.29577951);
14491:  'c2PI','Single').setExtended( 6.283185307);
14492:  'cPIdiv2','Single').setExtended( 1.570796326);
14493:  'cPIdiv4','Single').setExtended( 0.785398163);
14494:  'c3PIdiv4','Single').setExtended( 2.35619449);
14495:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14496:  'cInv360','Single').setExtended( 1 / 360);
14497:  'c180','Single').setExtended( 180);
14498:  'c360','Single').setExtended( 360);
14499:  'cOneHalf','Single').setExtended( 0.5);
14500:  'cLn10','Single').setExtended( 2.302585093);
14501:  {'MinSingle','Extended').setExtended( 1.5e-45);
14502:  'MaxSingle','Extended').setExtended( 3.4e+38);
14503:  'MinDouble','Extended').setExtended( 5.0e-324);
14504:  'MaxDouble','Extended').setExtended( 1.7e+308);
14505:  'MinExtended','Extended').setExtended( 3.4e-4932);
14506:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14507:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14508:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14509: end;
14510:
14511: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14512: begin
14513:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14514:   (FindClass('TOBJECT'),'TFaceGroups
14515:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14516:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14517:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14518:   SIRegister_TBaseMeshObject(CL);
14519:   (FindClass('TOBJECT'),'TSkeletonFrameList
14520:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14521:   SIRegister_TSkeletonFrame(CL);
14522:   SIRegister_TSkeletonFrameList(CL);
14523:   (FindClass('TOBJECT'),'TSkeleton
14524:   (FindClass('TOBJECT'),'TSkeletonBone
14525:   SIRegister_TSkeletonBoneList(CL);
14526:   SIRegister_TSkeletonRootBoneList(CL);
14527:   SIRegister_TSkeletonBone(CL);
14528:   (FindClass('TOBJECT'),'TSkeletonColliderList
14529:   SIRegister_TSkeletonCollider(CL);
14530:   SIRegister_TSkeletonColliderList(CL);
14531:   (FindClass('TOBJECT'),'TGLBaseMesh
14532:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14533:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14534:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14535:    +'QuaternionList; end
14536:   SIRegister_TSkeleton(CL);
14537:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14538:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14539:   SIRegister_TMeshObject(CL);
14540:   SIRegister_TMeshObjectList(CL);
14541:   //TMeshObjectListClass', 'class of TMeshObjectList
14542:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14543:   SIRegister_TMeshMorphTarget(CL);
14544:   SIRegister_TMeshMorphTargetList(CL);
14545:   SIRegister_TMorphableMeshObject(CL);
14546:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14547:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14548:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14549:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14550:   SIRegister_TSkeletonMeshObject(CL);
14551:   SIRegister_TFaceGroup(CL);
14552:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14553:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14554:   SIRegister_TFGVertexIndexList(CL);
14555:   SIRegister_TFGVertexNormalTexIndexList(CL);
14556:   SIRegister_TFGIndexTexCoordList(CL);
14557:   SIRegister_TFaceGroups(CL);
14558:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14559:   SIRegister_TVectorFile(CL);
14560:   //TVectorFileClass', 'class of TVectorFile
14561:   SIRegister_TGLGLSMVectorFile(CL);
14562:   SIRegister_TGLBaseMesh(CL);
14563:   SIRegister_TGLFreeForm(CL);
14564:   TGLActorOption', '( aoSkeletonNormalizeNormals )
```

```
14565:   TGLActorOptions', 'set of TGLActorOption
14566:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14567:   (FindClass('TOBJECT'),'TGLActor
14568:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14569:   SIRegister_TActorAnimation(CL);
14570:   TActorAnimationName', 'String
14571:   SIRegister_TActorAnimations(CL);
14572:   SIRegister_TGLBaseAnimationControler(CL);
14573:   SIRegister_TGLAnimationControler(CL);
14574:   TActorFrameInterpolation', '( afpNone, afpLinear )
14575:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14576:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14577:   SIRegister_TGLActor(CL);
14578:   SIRegister_TVectorFileFormat(CL);
14579:   SIRegister_TVectorFileFormatsList(CL);
14580:   (FindClass('TOBJECT'),'EInvalidVectorFile
14581:  Function GetVectorFileFormats : TVectorFileFormatsList
14582:  Function VectorFileFormatsFilter : String
14583:  Function VectorFileFormatsSaveFilter : String
14584:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14585:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14586:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14587: end;
14588:
14589: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14590: begin
14591:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14592:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14593:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14594:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14595:   SIRegister_TOleStream(CL);
14596:   (FindClass('TOBJECT'),'TConnectionPoints
14597:   TConnectionKind', '( ckSingle, ckMulti )
14598:   SIRegister_TConnectionPoint(CL);
14599:   SIRegister_TConnectionPoints(CL);
14600:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14601:   (FindClass('TOBJECT'),'TActiveXControlFactory
14602:   SIRegister_TActiveXControl(CL);
14603:   //TActiveXControlClass', 'class of TActiveXControl
14604:   SIRegister_TActiveXControlFactory(CL);
14605:   SIRegister_TActiveFormControl(CL);
14606:   SIRegister_TActiveForm(CL);
14607:   //TActiveFormClass', 'class of TActiveForm
14608:   SIRegister_TActiveFormFactory(CL);
14609:   (FindClass('TOBJECT'),'TPropertyPageImpl
14610:   SIRegister_TPropertyPage(CL);
14611:   //TPropertyPageClass', 'class of TPropertyPage
14612:   SIRegister_TPropertyPageImpl(CL);
14613:   SIRegister_TActiveXPropertyPage(CL);
14614:   SIRegister_TActiveXPropertyPageFactory(CL);
14615:   SIRegister_TCustomAdapter(CL);
14616:   SIRegister_TAdapterNotifier(CL);
14617:   SIRegister_IFontAccess(CL);
14618:   SIRegister_TFontAdapter(CL);
14619:   SIRegister_IPictureAccess(CL);
14620:   SIRegister_TPictureAdapter(CL);
14621:   SIRegister_TOleGraphic(CL);
14622:   SIRegister_TStringsAdapter(CL);
14623:   SIRegister_TReflectorWindow(CL);
14624:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14625:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14626:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14627:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14628:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14629:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14630:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14631:  Function ParkingWindow : HWND
14632: end;
14633:
14634: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14635: begin
14636:  // TIp6Bytes = array [0..15] of Byte;
14637: {:binary form of IPv6 adress (for string conversion routines)}
14638:  // TIp6Words = array [0..7] of Word;
14639:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14640:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14641:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14642:  Function synaIsIP( const Value : string) : Boolean');
14643:  Function synaIsIP6( const Value : string) : Boolean');
14644:  Function synaIPToID( Host : string) : Ansistring');
14645:  Function synaStrToIp6( value : string) : TIp6Bytes');
14646:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14647:  Function synaStrToIp( value : string) : integer');
14648:  Function synaIpToStr( value : integer) : string');
14649:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14650:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14651:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14652:  Function xStrToIP( const Value : String) : TIPAdr');
14653:  Function xIPToStr( const Adresse : TIPAdr) : String');
```

```
14654:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14655:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14656: end;
14657:
14658: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14659: begin
14660:  AddTypeS('TSpecials', 'set of Char');
14661:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14662:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14663:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14664:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14665:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14666:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14667:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14668:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14669:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14670:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14671:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14672:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14673:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
14674:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14675:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14676:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14677:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
14678:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14679:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14680:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14681:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14682:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14683:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14684:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14685:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14686:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
14687:  Function synCrc32( const Value : AnsiString) : Integer');
14688:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14689:  Function Crc16( const Value : AnsiString) : Word');
14690:  Function synMD5( const Value : AnsiString) : AnsiString');
14691:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14692:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14693:  Function synSHA1( const Value : AnsiString) : AnsiString');
14694:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
14695:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14696:  Function synMD4( const Value : AnsiString) : AnsiString');
14697: end;
14698:
14699: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14700: begin
14701:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14702:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14703:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14704:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14705:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14706:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14707:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14708:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14709:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14710:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14711:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14712:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14713:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14714:    +', CP864, CP865, CP869, CP1125 )');
14715:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14716:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14717:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
       TransformTable : array of Word) : AnsiString');
14718:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
       TransformTable : array of Word; Translit : Boolean) : AnsiString');
14719:  Function GetCurCP : TMimeChar');
14720:  Function GetCurOEMCP : TMimeChar');
14721:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14722:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14723:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14724:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14725:  Function GetBOM( Value : TMimeChar) : AnsiString');
14726:  Function StringToWide( const Value : AnsiString) : WideString');
14727:  Function WideToString( const Value : WideString) : AnsiString');
14728: end;
14729:
14730: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14731: begin
14732:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14733:  Procedure WakeOnLan( MAC, IP : string)');
14734:  Function GetDNS : string');
14735:  Function GetIEProxy( protocol : string) : TProxySetting');
14736:  Function GetLocalIPs : string');
14737: end;
14738:
14739:
14740: procedure SIRegister_synaser(CL: TPSPascalCompiler);
```

```
14741: begin
14742:  AddConstantN('synCR','Char #$0d);
14743:  Const('synLF','Char #$0a);
14744:  Const('cSerialChunk','LongInt'( 8192);
14745:  Const('LockfileDirectory','String '/var/lock');
14746:  Const('PortIsClosed','LongInt'( - 1);
14747:  Const('ErrAlreadyOwned','LongInt'( 9991);
14748:  Const('ErrAlreadyInUse','LongInt'( 9992);
14749:  Const('ErrWrongParameter','LongInt'( 9993);
14750:  Const('ErrPortNotOpen','LongInt'( 9994);
14751:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14752:  Const('ErrMaxBuffer','LongInt'( 9996);
14753:  Const('ErrTimeout','LongInt'( 9997);
14754:  Const('ErrNotRead','LongInt'( 9998);
14755:  Const('ErrFrame','LongInt'( 9999);
14756:  Const('ErrOverrun','LongInt'( 10000);
14757:  Const('ErrRxOver','LongInt'( 10001);
14758:  Const('ErrRxParity','LongInt'( 10002);
14759:  Const('ErrTxFull','LongInt'( 10003);
14760:  Const('dcb_Binary','LongWord')( $00000001);
14761:  Const('dcb_ParityCheck','LongWord')( $00000002);
14762:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14763:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14764:  Const('dcb_DtrControlMask','LongWord')( $00000030);
14765:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14766:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14767:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14768:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14769:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14770:  Const('dcb_OutX','LongWord')( $00000100);
14771:  Const('dcb_InX','LongWord')( $00000200);
14772:  Const('dcb_ErrorChar','LongWord')( $00000400);
14773:  Const('dcb_NullStrip','LongWord')( $00000800);
14774:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14775:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14776:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14777:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14778:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14779:  Const('dcb_AbortOnError','LongWord')( $00004000);
14780:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14781:  Const('synSB1','LongInt'( 0);
14782:  Const('SB1andHalf','LongInt'( 1);
14783:  Const('synSB2','LongInt'( 2);
14784:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14785:  Const('CS7fix','LongWord')( $0000020);
14786:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14787:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14788:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14789:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14790:   //AddTypeS('PDCB', '^TDCB // will not work');
14791:  //Const('MaxRates','LongInt'( 18);
14792:  //Const('MaxRates','LongInt'( 30);
14793:  //Const('MaxRates','LongInt'( 19);
14794:  Const('O_SYNC','LongWord')( $0080);
14795:  Const('synOK','LongInt'( 0);
14796:  Const('synErr','LongInt'( integer ( - 1 ));
14797:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
      HR_WriteCount, HR_Wait )');
14798:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14799:   SIRegister_ESynaSerError(CL);
14800:   SIRegister_TBlockSerial(CL);
14801:  Function GetSerialPortNames : string');
14802: end;
14803:
14804: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14805: begin
14806:  Const('DLLIconvName','String 'libiconv.so');
14807:  Const('DLLIconvName','String 'iconv.dll');
14808:   AddTypeS('size_t', 'Cardinal');
14809:   AddTypeS('iconv_t', 'Integer');
14810:   //AddTypeS('iconv_t', 'Pointer');
14811:   AddTypeS('argptr', 'iconv_t');
14812:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14813:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14814:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14815:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14816:  Function SynaIconvClose( var cd : iconv_t) : integer');
14817:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14818:  Function IsIconvloaded : Boolean');
14819:  Function InitIconvInterface : Boolean');
14820:  Function DestroyIconvInterface : Boolean');
14821:  Const('ICONV_TRIVIALP','LongInt'( 0);
14822:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14823:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14824:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14825:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14826: end;
14827:
14828: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
```

```
14829: begin
14830:  Const 'ICMP_ECHO','LongInt'( 8);
14831:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14832:  Const('ICMP_UNREACH','LongInt'( 3);
14833:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14834:  Const('ICMP6_ECHO','LongInt'( 128);
14835:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14836:  Const('ICMP6_UNREACH','LongInt'( 1);
14837:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14838:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14839:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14840:   SIRegister_TPINGSend(CL);
14841:  Function PingHost( const Host : string) : Integer');
14842:  Function TraceRouteHost( const Host : string) : string');
14843: end;
14844:
14845: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14846: begin
14847:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14848:  Const('synASN1_INT','LongWord')( $02);
14849:  Const('synASN1_OCTSTR','LongWord')( $04);
14850:  Const('synASN1_NULL','LongWord')( $05);
14851:  Const('synASN1_OBJID','LongWord')( $06);
14852:  Const('synASN1_ENUM','LongWord')( $0a);
14853:  Const('synASN1_SEQ','LongWord')( $30);
14854:  Const('synASN1_SETOF','LongWord')( $31);
14855:  Const('synASN1_IPADDR','LongWord')( $40);
14856:  Const('synASN1_COUNTER','LongWord')( $41);
14857:  Const('synASN1_GAUGE','LongWord')( $42);
14858:  Const('synASN1_TIMETICKS','LongWord')( $43);
14859:  Const('synASN1_OPAQUE','LongWord')( $44);
14860:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14861:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
14862:  Function synASNEncLen( Len : Integer) : AnsiString');
14863:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14864:  Function synASNEncInt( Value : Integer) : AnsiString');
14865:  Function synASNEncUInt( Value : Integer) : AnsiString');
14866:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14867:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14868:  Function synMibToId( Mib : String) : AnsiString');
14869:  Function synIdToMib( const Id : AnsiString) : String');
14870:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14871:  Function ASNdump( const Value : AnsiString) : AnsiString');
14872:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
14873:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14874: end;
14875:
14876: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14877: begin
14878:  Const('cLDAPProtocol','String '389');
14879:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14880:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14881:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14882:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14883:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14884:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14885:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14886:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14887:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14888:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14889:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14890:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14891:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14892:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14893:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14894:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14895:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
14896:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14897:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14898:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14899:   SIRegister_TLDAPAttribute(CL);
14900:   SIRegister_TLDAPAttributeList(CL);
14901:   SIRegister_TLDAPResult(CL);
14902:   SIRegister_TLDAPResultList(CL);
14903:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14904:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14905:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14906:   SIRegister_TLDAPSend(CL);
14907:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14908: end;
14909:
14910:
14911: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14912: begin
14913:  Const('cSysLogProtocol','String '514');
14914:  Const('FCL_Kernel','LongInt'( 0);
14915:  Const('FCL_UserLevel','LongInt'( 1);
14916:  Const('FCL_MailSystem','LongInt'( 2);
14917:  Const('FCL_System','LongInt'( 3);
```

```
14918:  Const('FCL_Security','LongInt'( 4);
14919:  Const('FCL_Syslogd','LongInt'( 5);
14920:  Const('FCL_Printer','LongInt'( 6);
14921:  Const('FCL_News','LongInt'( 7);
14922:  Const('FCL_UUCP','LongInt'( 8);
14923:  Const('FCL_Clock','LongInt'( 9);
14924:  Const('FCL_Authorization','LongInt'( 10);
14925:  Const('FCL_FTP','LongInt'( 11);
14926:  Const('FCL_NTP','LongInt'( 12);
14927:  Const('FCL_LogAudit','LongInt'( 13);
14928:  Const('FCL_LogAlert','LongInt'( 14);
14929:  Const('FCL_Time','LongInt'( 15);
14930:  Const('FCL_Local0','LongInt'( 16);
14931:  Const('FCL_Local1','LongInt'( 17);
14932:  Const('FCL_Local2','LongInt'( 18);
14933:  Const('FCL_Local3','LongInt'( 19);
14934:  Const('FCL_Local4','LongInt'( 20);
14935:  Const('FCL_Local5','LongInt'( 21);
14936:  Const('FCL_Local6','LongInt'( 22);
14937:  Const('FCL_Local7','LongInt'( 23);
14938:   Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
14939:   SIRegister_TSyslogMessage(CL);
14940:   SIRegister_TSyslogSend(CL);
14941:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
       Content:string):Boolean;
14942: end;
14943:
14944:
14945: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14946: begin
14947:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14948:   SIRegister_TMessHeader(CL);
14949:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14950:   SIRegister_TMimeMess(CL);
14951: end;
14952:
14953: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14954: begin
14955:   (FindClass('TOBJECT'),'TMimePart');
14956:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
14957:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14958:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14959:   SIRegister_TMimePart(CL);
14960:  Const('MaxMimeType','LongInt'( 25);
14961:  Function GenerateBoundary : string');
14962: end;
14963:
14964: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
14965: begin
14966:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
14967:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
14968:  Function NeedInline( const Value : AnsiString) : boolean');
14969:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
14970:  Function InlineCode( const Value : string) : string');
14971:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
14972:  Function InlineEmail( const Value : string) : string');
14973: end;
14974:
14975: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
14976: begin
14977:  Const('cFtpProtocol','String '21');
14978:  Const('cFtpDataProtocol','String '20');
14979:  Const('FTP_OK','LongInt'( 255);
14980:  Const('FTP_ERR','LongInt'( 254);
14981:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
14982:   SIRegister_TFTPListRec(CL);
14983:   SIRegister_TFTPList(CL);
14984:   SIRegister_TFTPSend(CL);
14985:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
14986:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
14987:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
       ToPort, ToFile, ToUser, ToPass : string) : Boolean');
14988: end;
14989:
14990: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
14991: begin
14992:  Const('cHttpProtocol','String '80');
14993:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
14994:   SIRegister_THTTPSend(CL);
14995:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
14996:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
14997:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
14998:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
14999:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
       ResultData:TStrings):Bool;
15000: end;
15001:
15002: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15003: begin
```

```
15004:  Const('cSmtpProtocol','String '25');
15005:   SIRegister_TSMTPSend(CL);
15006:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
       Passw:string):Bool;
15007:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15008:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
       Username, Password : string):Boolean');
15009: end;
15010:
15011: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15012: begin
15013:  Const('cSnmpProtocol','String '161');
15014:  Const('cSnmpTrapProtocol','String '162');
15015:  Const('SNMP_V1','LongInt'( 0);
15016:  Const('SNMP_V2C','LongInt'( 1);
15017:  Const('SNMP_V3','LongInt'( 3);
15018:  Const('PDUGetRequest','LongWord')( $A0);
15019:  Const('PDUGetNextRequest','LongWord')( $A1);
15020:  Const('PDUGetResponse','LongWord')( $A2);
15021:  Const('PDUSetRequest','LongWord')( $A3);
15022:  Const('PDUTrap','LongWord')( $A4);
15023:  Const('PDUGetBulkRequest','LongWord')( $A5);
15024:  Const('PDUInformRequest','LongWord')( $A6);
15025:  Const('PDUTrapV2','LongWord')( $A7);
15026:  Const('PDUReport','LongWord')( $A8);
15027:  Const('ENoError',LongInt 0;
15028:  Const('ETooBig','LongInt')( 1);
15029:  Const('ENoSuchName','LongInt'( 2);
15030:  Const('EBadValue','LongInt'( 3);
15031:  Const('EReadOnly','LongInt'( 4);
15032:  Const('EGenErr','LongInt'( 5);
15033:  Const('ENoAccess','LongInt'( 6);
15034:  Const('EWrongType','LongInt'( 7);
15035:  Const('EWrongLength','LongInt'( 8);
15036:  Const('EWrongEncoding','LongInt'( 9);
15037:  Const('EWrongValue','LongInt'( 10);
15038:  Const('ENoCreation','LongInt'( 11);
15039:  Const('EInconsistentValue','LongInt'( 12);
15040:  Const('EResourceUnavailable','LongInt'( 13);
15041:  Const('ECommitFailed','LongInt'( 14);
15042:  Const('EUndoFailed','LongInt'( 15);
15043:  Const('EAuthorizationError','LongInt'( 16);
15044:  Const('ENotWritable','LongInt'( 17);
15045:  Const('EInconsistentName','LongInt'( 18);
15046:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15047:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15048:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15049:   SIRegister_TSNMPMib(CL);
15050:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15051:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15052:   SIRegister_TSNMPRec(CL);
15053:   SIRegister_TSNMPSend(CL);
15054:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15055:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
15056:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15057:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15058:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15059:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15060:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : TStringList) : Integer');
15061: end;
15062:
15063: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15064: begin
15065:  Function GetDomainName2: AnsiString');
15066:  Function GetDomainController( Domain : AnsiString) : AnsiString');
15067:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15068:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15069:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15070:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15071: end;
15072:
15073: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15074: begin
15075:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15076:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15077:  Function wwStrToDate( const S : string) : boolean');
15078:  Function wwStrToTime( const S : string) : boolean');
15079:  Function wwStrToDateTime( const S : string) : boolean');
15080:  Function wwStrToTimeVal( const S : string) : TDateTime');
15081:  Function wwStrToDateVal( const S : string) : TDateTime');
15082:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15083:  Function wwStrToInt( const S : string) : boolean');
15084:  Function wwStrToFloat( const S : string) : boolean');
15085:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15086:  Function wwNextDay( Year, Month, Day : Word) : integer');
15087:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15088:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
```

```
15089:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15090:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15091:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15092:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15093:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15094:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15095:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15096: end;
15097:
15098: unit uPSI_Themes;
15099: Function ThemeServices : TThemeServices');
15100: Function ThemeControl( AControl : TControl) : Boolean');
15101: Function UnthemedDesigner( AControl : TControl) : Boolean');
15102: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15103: begin
15104:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15105: end;
15106: Unit uPSC_menus;
15107:  Function StripHotkey( const Text : string) : string');
15108:  Function GetHotkey( const Text : string) : string');
15109:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15110:  Function IsAltGRPressed : boolean');
15111:
15112: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15113: begin
15114:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15115:   SIRegister_TIdIMAP4Server(CL);
15116: end;
15117:
15118: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15119: begin
15120:  'HASH_SIZE','LongInt'( 256);
15121:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15122:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15123:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15124:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15125:    +' : Integer; Value : Variant; end');
15126:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15127:   SIRegister_TVariantSymbolTable(CL);
15128: end;
15129:
15130: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15131: begin
15132:   SIRegister_TThreadLocalVariables(CL);
15133:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15134:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15135:  Function ThreadLocals : TThreadLocalVariables');
15136:  Procedure WriteDebug( sz : String)');
15137:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15138:  'UDF_FAILURE','LongInt'( 1);
15139:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15140:  CL.AddTypeS('mTByteArray', 'array of byte;');
15141:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15142:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15143:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15144:  function IsNetworkConnected: Boolean;
15145:  function IsInternetConnected: Boolean;
15146:  function IsCOMConnected: Boolean;
15147:  function IsNetworkOn: Boolean;
15148:  function IsInternetOn: Boolean;
15149:  function IsCOMOn: Boolean;
15150:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15151:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15152:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15153:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15154:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15155:  Function GetMenu( hWnd : HWND) : HMENU');
15156:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15157: end;
15158:
15159: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15160: begin
15161:   SIRegister_IDataBlock(CL);
15162:   SIRegister_ISendDataBlock(CL);
15163:   SIRegister_ITransport(CL);
15164:   SIRegister_TDataBlock(CL);
15165:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15166:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15167:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15168:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15169:   SIRegister_TCustomDataBlockInterpreter(CL);
15170:   SIRegister_TSendDataBlock(CL);
15171:   'CallSig','LongWord')( $D800);
15172:   'ResultSig','LongWord')( $D400);
15173:   'asMask','LongWord')( $00FF);
15174:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15175:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15176:  Procedure CheckSignature( Sig : Integer)');
15177: end;
```

```
15178:
15179: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15180: begin
15181:   //CL.AddTypeS('HINTERNET', '___Pointer');
15182:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15183:   CL.AddTypeS('HINTERNET', 'Integer');
15184:   CL.AddTypeS('HINTERNET2', '___Pointer');
15185:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15186:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15187:   CL.AddTypeS('INTERNET_PORT', 'Word');
15188:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15189:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15190:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15191:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15192:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15193:  Function InternetCrackUrl(lpszUrl:PChar; dwUrlLength,dwFlags:DWORD; var
       lpUrlComponents:TURLComponents):BOOL;
15194:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
       dwUrlLength:DWORD):BOOL;
15195:  Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15196:  Function
       InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
       lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15197:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15198:  ;dwContext:DWORD):HINTERNET;
15199:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
       lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15200:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
       dwContext:DWORD):BOOL;
15201:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15202:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15203:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15204:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15205:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15206:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15207:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15208:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
       lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15209:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
       lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15210:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15211:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15212:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
       dwContext:DWORD):BOOL;
15213:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
       TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15214:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
       BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
15215:  Function
       WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15216:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
15217:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15218:  Function
       FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15219:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15220:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15221:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15222:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15223:  Function
       FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15224:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15225:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15226:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15227:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15228:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15229:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15230:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15231:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15232:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15233:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15234:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15235:  Function
       GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
       PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15236:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15237:  Function
       GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15238:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
       PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15239:  Function
       HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15240:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
       dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15241:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15242:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15243:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15244:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15245:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
```

```
15246:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15247:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15248:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
       lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15249:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
       lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15250:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15251:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15252:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15253:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15254: end;
15255:
15256: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15257: begin
15258:   AddTypeS('strCharSet', 'set of char');
15259:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15260:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15261:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15262:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
15263:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15264:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15265:  Procedure strStripWhiteSpace( var s : string)');
15266:  Function strRemoveChar( str : string; removeChar : char) : string');
15267:  Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15268:  Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15269:  Function wwEqualStr( s1, s2 : string) : boolean');
15270:  Function strCount( s : string; delimeter : char) : integer');
15271:  Function strWhiteSpace : strCharSet');
15272:  Function wwExtractFileNameOnly( const FileName : string) : string');
15273:  Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15274:  Function strTrailing( s : string; delimeter : char) : string');
15275:  Function strPreceding( s : string; delimeter : char) : string');
15276:  Function wwstrReplace( s, Find, Replace : string) : string');
15277: end;
15278:
15279: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15280: begin
15281:   SIRegister_TRemoteDataModule(CL);
15282:   SIRegister_TCRemoteDataModule(CL);
15283:  Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)');
15284:  Procedure UnregisterPooled( const ClassID : string)');
15285:  Procedure EnableSocketTransport( const ClassID : string)');
15286:  Procedure DisableSocketTransport( const ClassID : string)');
15287:  Procedure EnableWebTransport( const ClassID : string)');
15288:  Procedure DisableWebTransport( const ClassID : string)');
15289: end;
15290:
15291: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15292: begin
15293:  Function mxArcCos( x : Real) : Real');
15294:  Function mxArcSin( x : Real) : Real');
15295:  Function Comp2Str( N : Comp) : String');
15296:  Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15297:  Function Int2Str( N : LongInt) : String');
15298:  Function mxIsEqual( R1, R2 : Double) : Boolean');
15299:  Function LogXY( x, y : Real) : Real');
15300:  Function Pennies2Dollars( C : Comp) : String');
15301:  Function mxPower( X : Integer; Y : Integer) : Real');
15302:  Function Real2Str( N : Real; Width, Places : integer) : String');
15303:  Function mxStr2Comp( MyString : string) : Comp');
15304:  Function mxStr2Pennies( S : String) : Comp');
15305:  Function Str2Real( MyString : string) : Real');
15306:  Function XToTheY( x, y : Real) : Real');
15307: end;
15308:
15309: //************************Cindy Functions!*****************************
15310: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15311: begin
15312:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15313:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15314:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15315:  MessagePlainText','String 'text/plain');
15316:  CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15317:  MessageAlterText_Html','String 'multipart/alternative');
15318:  MessageHtml_Attach','String 'multipart/mixed');
15319:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15320:  MessageAlterText_Html_Attach','String 'multipart/mixed');
15321:  MessageAlterText_Html_RelatedAttach','String')('multipart/related;type="multipart/alternative"';
15322:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15323:  MessageReadNotification','String').( 'multipart/report; report-type="disposition-notification"';
15324:  Function ForceDecodeHeader( aHeader : String) : String');
15325:  Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string');
15326:  Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;');
15327:  Function Base64_DecodeToBytes( Value : String) : TBytes;');
15328:  Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15329:  Function Get_MD5( const aFileName : string) : string');
15330:  Function Get_MD5FromString( const aString : string) : string');
15331: end;
```

```
15332:
15333: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15334: begin
15335:  Function IsFolder( SRec : TSearchrec) : Boolean');
15336:  Function isFolderReadOnly( Directory : String) : Boolean');
15337:  Function DirectoryIsEmpty( Directory : String) : Boolean');
15338:  Function DirectoryWithSubDir( Directory : String) : Boolean');
15339:  Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15340:  Function DiskFreeBytes( Drv : Char) : Int64');
15341:  Function DiskBytes( Drv : Char) : Int64');
15342:  Function GetFileBytes( Filename : String) : Int64');
15343:  Function GetFilesBytes( Directory, Filter : String) : Int64');
15344:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15345:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15346:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15347:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15348:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15349:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15350:  SE_TCB_NAME','String 'SeTcbPrivilege');
15351:  SE_SECURITY_NAME','String 'SeSecurityPrivilege');
15352:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15353:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15354:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
15355:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15356:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15357:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15358:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15359:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15360:  SE_BACKUP_NAME','String 'SeBackupPrivilege');
15361:  SE_RESTORE_NAME','String 'SeRestorePrivilege');
15362:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15363:  SE_DEBUG_NAME','String 'SeDebugPrivilege');
15364:  SE_AUDIT_NAME','String 'SeAuditPrivilege');
15365:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15366:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15367:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15368:  SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15369:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15370:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
15371:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15372: end;
15373:
15374:
15375: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15376: begin
15377:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15378:    +'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15379:  Function ShellGetExtensionName( FileName : String) : String');
15380:  Function ShellGetIconIndex( FileName : String) : Integer');
15381:  Function ShellGetIconHandle( FileName : String) : HIcon');
15382:  Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15383:  Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
15384:  Procedure ShellRenameDir( DirFrom, DirTo : string)');
15385:  Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15386:  Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15387:  Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)');
15388:  Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)');
15389:  Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
       WaitCloseCompletion : boolean) : Boolean');
15390:  Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15391:  Function RemoveDuplicatedPathDelimiter( Str : String) : String');
15392:  Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime');
15393:  Function GetModificationDate( Filename : String) : TDateTime');
15394:  Function GetCreationDate( Filename : String) : TDateTime');
15395:  Function GetLastAccessDate( Filename : String) : TDateTime');
15396:  Function FileDelete( Filename : String) : Boolean');
15397:  Function FileIsOpen( Filename : string) : boolean');
15398:  Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15399:  Function DirectoryDelete( Directory : String) : Boolean');
15400:  Function GetPrinters( PrintersList : TStrings) : Integer');
15401:  Procedure SetDefaultPrinter( PrinterName : String)');
15402:  Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15403:  Function WinToDosPath( WinPathName : String) : String');
15404:  Function DosToWinPath( DosPathName : String) : String');
15405:  Function cyGetWindowsVersion : TWindowsVersion');
15406:  Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean');
15407:  Procedure WindowsShutDown( Restart : boolean)');
15408:  Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
       FileIcon:string;NumIcone:integer);
15409:  Procedure GetWindowsFonts( FontsList : TStrings)');
15410:  Function GetAvailableFilename( DesiredFileName : String): String');
15411: end;
15412:
15413: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15414: begin
15415:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15416:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15417:   Type(TStringRead', '( srFromLeft, srFromRight )');
15418:   Type(TStringReads', 'set of TStringRead');
```

```
15419:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15420:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15421:   Type(TWordsOptions', 'set of TWordsOption');
15422:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15423:   Type(TCarTypes', 'set of TCarType');
15424:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15425:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15426:  Function Char_GetType( aChar : Char) : TCarType');
15427:  Function SubString_Count( Str : String; Separator : Char) : Integer');
15428:  Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15429:  Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String');
15430:  Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15431:  Procedure SubString_Add( var Str : String; Separator : Char; Value : String)');
15432:  Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String)');
15433:  Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String)');
15434:  Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean');
15435:  Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15436:  Function SubString_Ribbon( Str : string; Separator : Char; Current: Word; MoveBy : Integer):Integer;');
15437:  Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String;');
15438:  Function String_Quote( Str : String) : String');
15439:  Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char');
15440:  Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15441:  Function String_GetWord( Str : String; StringRead : TStringRead) : String');
15442:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
15443:  Function String_ToInt( Str : String) : Integer');
15444:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15445:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15446:  Function String_Reverse( Str : String) : String');
15447:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15448:  Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
       TCaseSensitive):Integer;');
15449:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15450:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15451:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
       Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15452:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15453:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15454:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15455:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15456:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15457:  Function String_End( Str : String; Cars : Word) : String');
15458:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
       AlwaysFindFromBeginning : Boolean) : String');
15459:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15460:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15461:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15462:  Function String_IsNumbers( Str : String) : Boolean');
15463:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15464:  Function StringToCsvCell( aStr : String) : String');
15465: end;
15466:
15467: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15468: begin
15469:  Function LongDayName( aDate : TDate) : String');
15470:  Function LongMonthName( aDate : TDate) : String');
15471:  Function ShortYearOf( aDate : TDate) : byte');
15472:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15473:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15474:  Function SecondsToMinutes( Seconds : Integer) : Double');
15475:  Function MinutesToSeconds( Minutes : Double) : Integer');
15476:  Function MinutesToHours( Minutes : Integer) : Double');
15477:  Function HoursToMinutes( Hours : Double) : Integer');
15478:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15479:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15480:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15481:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15482:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15483:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15484:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
       RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15485:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15486:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15487: end;
15488:
15489: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15490: begin
15491:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15492:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15493:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15494:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15495:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15496:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15497:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15498:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15499:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15500:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15501:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15502:  Function
       TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
```

```
15503:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15504:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
        CopySubChildren:Bool;
15505:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15506:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15507:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15508:  Procedure cyCenterControl( aControl : TControl)');
15509:  Function GetLastParent( aControl : TControl) : TWinControl');
15510:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15511:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15512: end;
15513:
15514: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15515: begin
15516:  Function TablePackTable( Tab : TTable) : Boolean');
15517:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15518:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15519:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15520:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15521:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
15522:  Function TableEmptyTable( Tab : TTable) : Boolean');
15523:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15524:  Procedure TableFindNearest( aTable : TTable; Value : String)');
15525:  Function
        TableCreate(Owner:TComponent;DataBaseName:ShortString;TableName:String;IndexName:ShortString;ReadOnly :
        Boolean):TTable;
15526:  Function
        TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15527:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15528: end;
15529:
15530: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15531: begin
15532:   SIRegister_TcyRunTimeDesign(CL);
15533:   SIRegister_TcyShadowText(CL);
15534:   SIRegister_TcyBgPicture(CL);
15535:   SIRegister_TcyGradient(CL);
15536:   SIRegister_tcyBevel(CL);
15537:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15538:   SIRegister_tcyBevels(CL);
15539:   SIRegister_TcyImagelistOptions(CL);
15540:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15541: end;
15542:
15543: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15544: begin
15545:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
        adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
        Maxdegrade : Byte;'+
15546:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
15547:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15548:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15549:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
        Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
15550:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
        Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15551:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15552:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15553:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
        BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15554:  Procedure cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,
        BottomRightColor:TColor;Width:Integer;const DrawLeft:Boolean;const DrawTop:Bool;const DrawRight:Bool;const
        DrawBottom:Bool;const RoundRect:bool;
15555:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
        aState : TButtonState; Focused, Hot : Boolean)');
15556:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
        aState : TButtonState; Focused, Hot : Boolean)');
15557:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
        TColor; aState : TButtonState; Focused, Hot : Boolean)');
15558:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
        GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15559:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
        const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15560:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
        TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15561:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
        TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15562:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
        WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15563:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15564:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15565:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15566:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
        CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15567:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15568:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15569:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
```

```
15570:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15571:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15572:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15573:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15574:  Procedure DrawCanvas1(Destination:TCanvas;
        DestRect:TRect;Src:TCanvas;SrcRect:TRect;TransparentColor:TColor; const aStyle:TBgStyle; const
        aPosition:TBgPosition; const IndentX : Integer; const IndentY : Integer;const IntervalX : Integer; const
        IntervalY : Integer; const RepeatX:Integer;const RepeatY:Integer);
15575:  Procedure DrawGraphic( Destination : TCanvas; DestRect : TRect; aGraphic : TGraphic; SrcRect : TRect;
        TransparentColor:TColor; const aStyle : TBgStyle; const aPosition : TBgPosition; const IndentX : Integer;
        const IndentY:Integer;const IntervalX:Integer;const IntervalY:Integer;const RepeatX:Integer;const
        RepeatY:Integer;
15576:  Procedure DrawGraphic1(Destination:TCanvas; DestRect:TRect;aGraphic : TGraphic; Transparent : Boolean;
        const aStyle : TBgStyle; const aPosition : TBgPosition; const IndentX : Integer; const IndentY : Integer;
        const  IntervalX:Integer; const IntervalY : Integer; const RepeatX:Integer; const RepeatY:Integer);');
15577:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15578:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15579:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15580:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15581:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
15582:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15583:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15584:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15585:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
15586:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15587:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
15588:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15589:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15590:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15591:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15592:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15593: end;
15594:
15595: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15596: begin
15597:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15598:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15599:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15600:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15601:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15602:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15603:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15604:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15605:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15606:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15607:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15608:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
        bmInvertReverseFromColor);
15609:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
        rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15610:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15611:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15612: end;
15613:
15614: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15615: begin
15616:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15617:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15618:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15619:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15620:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15621:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15622:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15623:  Const SC_GROUP_IDENTIFIERW','String') '+');
15624:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15625:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15626:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15627:  Const SERVICE_INACTIVE','LongWord $00000002);
15628:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
15629:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15630:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15631:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15632:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15633:  Const SERVICE_STOPPED','LongWord $00000001);
15634:  Const SERVICE_START_PENDING','LongWord $00000002);
15635:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15636:  Const SERVICE_RUNNING','LongWord $00000004);
15637:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15638:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15639:  Const SERVICE_PAUSED','LongWord $00000007);
15640:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15641:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15642:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15643:  Const SC_MANAGER_CONNECT','LongWord $0001);
15644:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15645:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15646:  Const SC_MANAGER_LOCK','LongWord $0008);
15647:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15648:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
```

```
15649:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15650:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15651:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15652:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15653:  Const SERVICE_START','LongWord $0010);
15654:  Const SERVICE_STOP','LongWord $0020);
15655:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15656:  Const SERVICE_INTERROGATE','LongWord $0080);
15657:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15658:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15659:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15660:  Const SERVICE_ADAPTER','LongWord $00000004);
15661:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15662:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15663:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15664:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15665:  Const SERVICE_BOOT_START','LongWord $00000000);
15666:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15667:  Const SERVICE_AUTO_START','LongWord $00000002);
15668:  Const SERVICE_DEMAND_START','LongWord $00000003);
15669:  Const SERVICE_DISABLED','LongWord $00000004);
15670:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15671:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15672:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
15673:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15674:   CL.AddTypeS('SC_HANDLE', 'THandle');
15675:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15676:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15677:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15678:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15679:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15680:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15681:   Const TServiceStatus', '_SERVICE_STATUS');
15682:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15683:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15684:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15685:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15686:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15687:   TEnumServiceStatus', 'TEnumServiceStatusA');
15688:   SC_LOCK', '___Pointer');
15689:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15690:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15691:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15692:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15693:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15694:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15695:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15696:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15697:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15698:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15699:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15700:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15701:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15702:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15703:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15704:   TQueryServiceConfig', 'TQueryServiceConfigA');
15705:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15706:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15707:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
       dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15708:   +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15709:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
       dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15710:   +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15711:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15712:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
       TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15713:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
       lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
       : DWORD):BOOL');
15714:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
       lpcchBuffer:DWORD):BOOL');
15715:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
       lpcchBuffer:DWORD):BOOL;
15716:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15717:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15718:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15719:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15720:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
       cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15721:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15722:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15723:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15724:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15725: end;
15726:
15727: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15728: begin
15729:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
       AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
       BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
```

```
15730:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
        DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15731:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15732:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
        TWinControl;
15733:  Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
        AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15734:  Function CreateNotifyThread(const
        FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15735: end;
15736:
15737: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15738: begin
15739:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15740:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15741:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15742:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15743:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
15744:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15745:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15746:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15747:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
15748:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15749:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
15750:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15751:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15752:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15753:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15754:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15755:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15756:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
15757:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15758:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15759:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15760:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15761:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15762:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15763:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15764:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15765:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15766:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15767:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
15768:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15769:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15770:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15771:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15772:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15773:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15774:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15775:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15776:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15777:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
15778:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15779:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15780:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15781:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15782:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15783:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15784:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
        List:TStrings):Bool
15785:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15786:   FindClass('TOBJECT'),'EJclFileSummaryError');
15787:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15788:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15789:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15790:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15791:   SIRegister_TJclFilePropertySet(CL);
15792:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15793:   SIRegister_TJclFileSummary(CL);
15794:   SIRegister_TJclFileSummaryInformation(CL);
15795:   SIRegister_TJclDocSummaryInformation(CL);
15796:   SIRegister_TJclMediaFileSummaryInformation(CL);
15797:   SIRegister_TJclMSISummaryInformation(CL);
15798:   SIRegister_TJclShellSummaryInformation(CL);
15799:   SIRegister_TJclStorageSummaryInformation(CL);
15800:   SIRegister_TJclImageSummaryInformation(CL);
15801:   SIRegister_TJclDisplacedSummaryInformation(CL);
15802:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15803:   SIRegister_TJclMiscSummaryInformation(CL);
15804:   SIRegister_TJclWebViewSummaryInformation(CL);
15805:   SIRegister_TJclMusicSummaryInformation(CL);
15806:   SIRegister_TJclDRMSummaryInformation(CL);
15807:   SIRegister_TJclVideoSummaryInformation(CL);
15808:   SIRegister_TJclAudioSummaryInformation(CL);
15809:   SIRegister_TJclControlPanelSummaryInformation(CL);
15810:   SIRegister_TJclVolumeSummaryInformation(CL);
15811:   SIRegister_TJclShareSummaryInformation(CL);
15812:   SIRegister_TJclLinkSummaryInformation(CL);
15813:   SIRegister_TJclQuerySummaryInformation(CL);
```

```
15814:   SIRegister_TJclImageInformation(CL);
15815:   SIRegister_TJclJpegSummaryInformation(CL);
15816: end;
15817:
15818: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15819: begin
15820:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15821:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15822:   T8087Infinity','( icProjective, icAffine )');
15823:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15824:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15825:  Function Get8087ControlWord : Word');
15826:  Function Get8087Infinity : T8087Infinity');
15827:  Function Get8087Precision : T8087Precision');
15828:  Function Get8087Rounding : T8087Rounding');
15829:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15830:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15831:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15832:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15833:  Function Set8087ControlWord( const Control : Word) : Word');
15834:  Function ClearPending8087Exceptions : T8087Exceptions');
15835:  Function GetPending8087Exceptions : T8087Exceptions');
15836:  Function GetMasked8087Exceptions : T8087Exceptions');
15837:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15838:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15839:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15840: end;
15841:
15842: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15843: begin
15844:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15845:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15846:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
15847:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15848:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15849:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15850:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
15851:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15852: end;
15853:
15854: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15855: begin
15856:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15857:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15858:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15859:   type ULONG', 'Cardinal');
15860:      LPCWSTR', 'PChar');
15861:   CL.AddTypeS('LPWSTR', 'PChar');
15862:   LPSTR', 'PChar');
15863:   TBindVerb', 'ULONG');
15864:   TBindInfoF', 'ULONG');
15865:   TBindF', 'ULONG');
15866:   TBSCF', 'ULONG');
15867:   TBindStatus', 'ULONG');
15868:   TCIPStatus', 'ULONG');
15869:   TBindString', 'ULONG');
15870:   TPiFlags', 'ULONG');
15871:   TOIBdgFlags', 'ULONG');
15872:   TParseAction', 'ULONG');
15873:   TPSUAction', 'ULONG');
15874:   TQueryOption', 'ULONG');
15875:   TPUAF', 'ULONG');
15876:   TSZMFlags', 'ULONG');
15877:   TUrlZone', 'ULONG');
15878:   TUrlTemplate', 'ULONG');
15879:   TZAFlags', 'ULONG');
15880:   TUrlZoneReg', 'ULONG');
15881:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15882:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15883:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15884:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15885:  const 'CF_NULL','LongInt').SetInt( 0);
15886:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15887:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15888:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15889:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15890:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15891:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15892:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15893:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15894:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15895:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15896:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15897:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15898:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15899:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15900:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15901:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
```

```
15902:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15903:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15904:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15905:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15906:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15907:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15908:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15909:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15910:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15911:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15912:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15913:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15914:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15915:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15916:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15917:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15918:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15919:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
15920:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15921:   SIRegister_IPersistMoniker(CL);
15922:   SIRegister_IBindProtocol(CL);
15923:   SIRegister_IBinding(CL);
15924:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15925:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15926:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15927:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15928:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15929:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15930:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15931:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
15932:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
15933:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
15934:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
15935:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
15936:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
15937:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
15938:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
15939:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
15940:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
15941:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
15942:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
15943:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
15944:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
15945:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
15946:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
15947:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
15948:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
15949:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
15950:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
15951:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
15952:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
15953:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
15954:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
15955:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
15956:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
15957:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
15958:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
15959:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
15960:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
15961:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
15962:  const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
15963:  const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
15964:  const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
15965:  const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
15966:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
15967:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
15968:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
15969:  const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
15970:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
15971:  const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
15972:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
15973:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
15974:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
15975:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
15976:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
15977:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
15978:  const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
15979:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
15980:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
15981:  const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
15982:  const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
15983:  const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
15984:  const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
15985:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
15986:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
15987:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
15988:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
15989:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
15990:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
```

```
15991:  const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED + 1);
15992:  const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
15993:  const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
15994:  const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
15995:  const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
15996:  const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
15997:  const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
15998:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
15999:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16000:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16001:  const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
16002:  const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED + 1);
16003:  const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16004:  const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
16005:  const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
16006:  const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
16007:  const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16008:  // PBindInfo', '^TBindInfo // will not work');
16009:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16010:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16011:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16012:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16013:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16014:   TBindInfo', '_tagBINDINFO');
16015:   BINDINFO', '_tagBINDINFO');}
16016:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16017:    +'criptor : DWORD; bInheritHandle : BOOL; end');
16018:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
16019:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
16020:   //PRemBindInfo', '^TRemBindInfo // will not work');
16021:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16022:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16023:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16024:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16025:    +'n; dwReserved : DWORD; end');
16026:   TRemBindInfo', '_tagRemBINDINFO');
16027:   RemBINDINFO', '_tagRemBINDINFO');}
16028:   //PRemFormatEtc', '^TRemFormatEtc // will not work');
16029:   tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16030:   TRemFormatEtc', 'tagRemFORMATETC');
16031:   RemFORMATETC', 'tagRemFORMATETC');
16032:   SIRegister_IBindStatusCallback(CL);
16033:   SIRegister_IAuthenticate(CL);
16034:   SIRegister_IHttpNegotiate(CL);
16035:   SIRegister_IWindowForBindingUI(CL);
16036:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16037:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16038:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16039:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16040:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16041:  const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16042:  const 'CIP_EXE_SELF_REGISTERATION_TIMEOUT',LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
16043:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16044:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16045:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16046:   SIRegister_ICodeInstall(CL);
16047:   SIRegister_IWinInetInfo(CL);
16048:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16049:   SIRegister_IHttpSecurity(CL);
16050:   SIRegister_IWinInetHttpInfo(CL);
16051:   SIRegister_IBindHost(CL);
16052:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16053:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16054:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16055:  Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback) : HResult');
16056:  Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback) : HResult');
16057:  Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
       : IBindStatusCallback) : HResult');
16058:  Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
       IBindStatusCallback) : HResult');
16059:  Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
       p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult');
16060:  Function HlinkGoBack( unk : IUnknown) : HResult');
16061:  Function HlinkGoForward( unk : IUnknown) : HResult');
16062:  Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult');
16063: // Function HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult');
16064:   SIRegister_IInternet(CL);
16065:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16066:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16067:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16068:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16069:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16070:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16071:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16072:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16073:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16074:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16075:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16076:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
```

```
16077:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16078:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16079:   //POLEStrArray', '^TOLESTRArray // will not work');
16080:   SIRegister_IInternetBindInfo(CL);
16081:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16082:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16083:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16084:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16085:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16086:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16087:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16088:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16089:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16090:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16091:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16092:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16093:   //PProtocolData', '^TProtocolData // will not work');
16094:   _tagPROTOCOLDATA','record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16095:   TProtocolData', '_tagPROTOCOLDATA');
16096:   PROTOCOLDATA', '_tagPROTOCOLDATA');
16097:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink, 'IInternetProtocolSink');
16098:   SIRegister_IInternetProtocolRoot(CL);
16099:   SIRegister_IInternetProtocol(CL);
16100:   SIRegister_IInternetProtocolSink(CL);
16101:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16102:   SIRegister_IInternetSession(CL);
16103:   SIRegister_IInternetThreadSwitch(CL);
16104:   SIRegister_IInternetPriority(CL);
16105:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16106:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16107:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16108:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16109:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16110:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16111:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16112:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16113:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16114:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16115:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16116:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16117:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16118:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16119:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16120:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16121:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16122:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16123:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16124:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16125:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16126:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16127:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16128:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16129:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16130:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16131:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16132:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16133:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16134:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16135:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16136:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16137:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16138:   SIRegister_IInternetProtocolInfo(CL);
16139:   IOInet', 'IInternet');
16140:   IOInetBindInfo', 'IInternetBindInfo');
16141:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16142:   IOInetProtocol', 'IInternetProtocol');
16143:   IOInetProtocolSink', 'IInternetProtocolSink');
16144:   IOInetProtocolInfo', 'IInternetProtocolInfo');
16145:   IOInetSession', 'IInternetSession');
16146:   IOInetPriority', 'IInternetPriority');
16147:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16148:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16149:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16150:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16151:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16152:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
       pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16153:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
       IInternetSes;dwReserved:DWORD):HResult;
16154:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16155:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16156:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16157:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16158:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
```

```
16159:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
       TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16160:  Function OInetGetSession(dwSessionMode:DWORD; var
       pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16161:    //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16162:   //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16163: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16164: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16165:   //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16166:   //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16167:   //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16168:   const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16169:    SIRegister_IInternetSecurityMgrSite(CL);
16170:   const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16171:   const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16172:   const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16173:   const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16174:   const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16175:   const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16176:   const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16177:   const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16178:   const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16179:   const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16180:   const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16181:   const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16182:   const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16183:   const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16184:   const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16185:   const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16186:   const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16187:   const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16188:   const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16189:   const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16190:   const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16191:   const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16192:   const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16193:   const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16194:   const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16195:   const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16196:   const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16197:    SIRegister_IInternetSecurityManager(CL);
16198:    SIRegister_IInternetHostSecurityManager(CL);
16199:    SIRegister_IInternetSecurityManagerEx(CL);
16200:   const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16201:   const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16202:   const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16203:   const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16204:   const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16205:   const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16206:   const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16207:   const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16208:   const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16209:   const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16210:   const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16211:   const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16212:   const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16213:   const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16214:   const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16215:   const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16216:   const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16217:   const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16218:   const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16219:   const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16220:   const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16221:   const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16222:   const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16223:   const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16224:   const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16225:   const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16226:   const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16227:   const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16228:   const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16229:   const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16230:   const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16231:   const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16232:   const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16233:   const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16234:   const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16235:   const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16236:   const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16237:   const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16238:   const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16239:   const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16240:   const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16241:   const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16242:   const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16243:   const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16244:   const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16245:   const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
```

```
16246:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16247:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16248:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16249:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16250:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16251:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16252:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16253:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16254:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16255:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16256:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16257:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16258:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16259:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16260:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16261:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16262:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16263:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16264:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16265:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16266:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16267:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16268:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16269:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16270:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16271:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16272:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16273:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16274:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16275:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16276:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16277:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16278:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16279:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16280:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16281:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16282:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16283:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16284:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16285:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16286:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16287:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16288:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16289:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16290:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16291:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16292:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16293:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16294:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16295:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16296:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16297:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16298:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16299:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16300:  Function GetUrlPolicyPermissions( dw : DWORD) : DWORD');
16301:  Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD');
16302:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16303:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16304:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16305:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16306:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16307:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16308:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16309:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16310:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16311:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16312:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16313:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16314:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16315:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16316:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16317:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16318:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16319:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16320:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16321:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16322:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16323:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16324:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16325:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16326:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16327:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16328:   //PZoneAttributes', '^TZoneAttributes // will not work');
16329:  _ZONEATTRIBUTES','record cbSize:ULONG;szDisplayName:array [0..260-1] of Char;szDescription:array [0..200 -
        1] of Char;szIconPath: array [0..260 - 1] of char; dwTemplateMinLevel: DWORD;  dwTemplateRecommended:
       DWORD;  dwTemplateCurrentLevel: DWORD;  dwFlags: DWORD; end');
16330:   { _ZONEATTRIBUTES = packed record
16331:     cbSize: ULONG;
16332:     szDisplayName: array [0..260 - 1] of WideChar;
```

```
16333:     szDescription: array [0..200 - 1] of WideChar;
16334:     szIconPath: array [0..260 - 1] of WideChar;
16335:     dwTemplateMinLevel: DWORD;
16336:     dwTemplateRecommended: DWORD;
16337:     dwTemplateCurrentLevel: DWORD;
16338:     dwFlags: DWORD;
16339:   end; }
16340:   TZoneAttributes', '_ZONEATTRIBUTES');
16341:   ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16342:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16343:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16344:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16345:   SIRegister_IInternetZoneManager(CL);
16346:   SIRegister_IInternetZoneManagerEx(CL);
16347:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16348:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16349:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16350:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16351:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16352:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16353:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16354:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
16355:   //PCodeBaseHold', '^TCodeBaseHold // will not work');
16356:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16357:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16358:   TCodeBaseHold', '_tagCODEBASEHOLD');
16359:   CODEBASEHOLD', '_tagCODEBASEHOLD');
16360:   //PSoftDistInfo', '^TSoftDistInfo // will not work');
16361:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
16362:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
16363:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
16364:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
16365:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16366:   TSoftDistInfo', '_tagSOFTDISTINFO');
16367:   SOFTDISTINFO', '_tagSOFTDISTINFO');
16368:   SIRegister_ISoftDistExt(CL);
16369:  Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16370:  Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
      dwAdvertisedVersionLS : DWORD) : HResult');
16371:   SIRegister_IDataFilter(CL);
16372:  // PProtocolFilterData', '^TProtocolFilterData // will not work');
16373:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
16374:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16375:    +'terFlags : DWORD; end');
16376:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16377:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16378:   //PDataInfo', '^TDataInfo // will not work');
16379:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
16380:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16381:   TDataInfo', '_tagDATAINFO');
16382:   DATAINFO', '_tagDATAINFO');
16383:   SIRegister_IEncodingFilterFactory(CL);
16384:  Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16385:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16386:  //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16387:   //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16388:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16389:    +rlName : LPSTR; StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16390:   THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16391:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16392:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16393: end;
16394:
16395: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16396: begin
16397:  Procedure reformatMemo( const m : TCustomMemo)');
16398:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16399:  Procedure MoveToTop( memo : TMemo)');
16400:  Procedure ScrollToTop( memo : TMemo)');
16401:  Function LineNumberClicked( memo : TMemo) : integer');
16402:  Function MemoClickedLine( memo : TMemo) : integer');
16403:  Function ClickedMemoLine( memo : TMemo) : integer');
16404:  Function MemoLineClicked( memo : TMemo) : integer');
16405:  Function LinePositionClicked( Memo : TMemo) : integer');
16406:  Function ClickedMemoPosition( memo : TMemo) : integer');
16407:  Function MemoPositionClicked( memo : TMemo) : integer');
16408:  Procedure AdjustGridSize( grid : TDrawGrid)');
16409:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16410:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16411:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16412:  Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');
16413:  Procedure sortstrDown( var s : string)');
16414:  Procedure sortstrUp( var s : string)');
16415:  Procedure rotatestrleft( var s : string)');
16416:  Function dffstrtofloatdef( s : string; default : extended) : extended');
16417:  Function deblank( s : string) : string');
16418:  Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16419:  Procedure FreeAndClearListBox( C : TListBox);');
16420:  Procedure FreeAndClearMemo( C : TMemo);');
```

```
16421:  Procedure FreeAndClearStringList( C : TStringList);');
16422:  Function dffgetfilesize( f : TSearchrec) : int64');
16423: end;
16424:
16425: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16426: begin
16427:   CL.AddTypeS('intset', 'set of byte');
16428:   TPoint64', 'record x : int64; y : int64; end');
16429:  Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16430:  Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16431:  Function GeneratePentagon( n : integer) : integer');
16432:  Function IsPentagon( p : integer) : boolean');
16433:  Function isSquare( const N : int64) : boolean');
16434:  Function isCube( const N : int64) : boolean');
16435:  Function isPalindrome( const n : int64) : boolean;');
16436:  Function isPalindrome1( const n : int64; var len : integer) : boolean;');
16437:  Function GetEulerPhi( n : int64) : int64');
16438:  Function dffIntPower( a, b : int64) : int64;');
16439:  Function IntPower1( a : extended; b : int64) : extended;');
16440:  Function gcd2( a, b : int64) : int64');
16441:  Function GCDMany( A : array of integer) : integer');
16442:  Function LCMMany( A : array of integer) : integer');
16443:  Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16444:  Function dffFactorial( n : int64) : int64');
16445:  Function digitcount( n : int64) : integer');
16446:  Function nextpermute( var a : array of integer) : boolean');
16447:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16448:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16449:  Function InttoBinaryStr( nn : integer) : string');
16450:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16451:  Function AngleToStr( angle : extended) : string');
16452:  Function deg2rad( deg : extended) : extended');
16453:  Function rad2deg( rad : extended) : extended');
16454:  Function GetLongToMercProjection( const long : extended) : extended');
16455:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16456:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16457:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16458:   SIRegister_TPrimes(CL);
16459:   //RIRegister_TPrimes(CL);
16460:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16461:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16462:  Function Random64( const N : Int64) : Int64;');
16463:  Procedure Randomize64');
16464:  Function Random641 : extended;');
16465:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16466: end;
16467:
16468: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16469: begin
16470:   TrealPoint', 'record x : extended; y : extended; end');
16471:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16472:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16473:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16474:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16475:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16476:  Function realpoint( x, y : extended) : TRealPoint');
16477:  Function dist( const p1, p2 : TrealPoint) : extended');
16478:  Function intdist( const p1, p2 : TPoint) : integer');
16479:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16480:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16481:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16482:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16483:  Function GetTheta( const L : TLine) : extended;');
16484:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16485:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16486:  Procedure Extendline( var L : TLine; dist : integer);');
16487:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16488:  Function Linesintersect( line1, line2 : TLine) : boolean');
16489:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16490:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16491:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16492:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16493:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16494:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16495:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16496:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16497:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16498:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16499:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16500:  Function DegtoRad( d : extended) : extended');
16501:  Function RadtoDeg( r : extended) : extended');
16502:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16503:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16504:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16505:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16506:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
```

```
16507:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16508:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16509:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16510:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16511: end;
16512:
16513:
16514: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16515: begin
16516:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16517:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16518:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16519:   TSunrec', 'record TrueEclLon:extended;
      AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
      TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16520:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16521:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16522:    +'arth : extended; Phase : extended; end');
16523:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16524:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16525:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16526:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
16527:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16528:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16529:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16530:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16531:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16532:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
      extended; UncorrectedEarthDistance : extended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended;
      ApparentRaDecl:TRPoint; end');
16533:   SIRegister_TAstronomy(CL);
16534:  Function AngleToStr( angle : extended) : string');
16535:  Function StrToAngle( s : string; var angle : extended) : boolean');
16536:  Function HoursToStr24( t : extended) : string');
16537:  Function RPoint( x, y : extended) : TRPoint');
16538:  Function getStimename( t : TDTType) : string');
16539: end;
16540:
16541: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16542: begin
16543:   TCardValue', 'Integer');
16544:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16545:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16546:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16547:   SIRegister_TCard(CL);
16548:   SIRegister_TDeck(CL);
16549: end;
16550:
16551: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16552: begin
16553:   tMethodCall', 'Procedure');
16554:   tVerboseCall', 'Procedure ( s : string)');
16555:  // PTEdge', '^TEdge // will not work');
16556:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16557:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16558:   SIRegister_TNode(CL);
16559:   SIRegister_TGraphList(CL);
16560: end;
16561:
16562: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16563: begin
16564:   ParserFloat', 'extended');
16565:   //PParserFloat', '^ParserFloat // will not work');
16566:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16567:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16568:   //POperation', '^TOperation // will not work');
16569:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16570:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16571:    +'; Token : TDFFToken; end');
16572:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16573:   (CL.FindClass('TOBJECT'),'EMathParserError');
16574:   CL.FindClass('TOBJECT'),'ESyntaxError');
16575:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16576:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16577:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16578:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16579:   (CL.FindClass('TOBJECT'),'EBadName');
16580:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16581:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16582:   SIRegister_TCustomParser(CL);
16583:   SIRegister_TExParser(CL);
16584: end;
16585:
16586:   function isService: boolean;
16587:   begin
16588:     result:= NOT(Application is TApplication);
16589:     {result:= Application is TServiceApplication;}
16590:   end;
16591:   function isApplication: boolean;
```

```
16592:   begin
16593:     result:= Application is TApplication;
16594:   end;
16595:   //SM_REMOTESESSION = $1000
16596:   function isTerminalSession: boolean;
16597:   begin
16598:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16599:   end;
16600:
16601: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16602: begin
16603:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16604:    +'String; margin_bottom : String; margin_left : String; margin_right : Strin'
16605:    +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16606:  Function cyURLEncode( const S : string) : string');
16607:  Function MakeResourceURL(const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16608:  Function MakeResourceURL1(const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;;
16609:  Function cyColorToHtml( aColor : TColor) : String');
16610:  Function HtmlToColor( aHtmlColor : String) : TColor');
16611:  //Function GetStreamEncoding( aStream : TStream) : TEncoding');
16612:  // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) : Boolean');
16613:  Function AddHtmlUnicodePrefix( aHtml : String) : String');
16614:  Function RemoveHtmlUnicodePrefix( aHtml : String) : String');
16615:  Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
16616:  Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16617:  CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16618:  CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16619:  CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16620:  CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16621:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16622:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
16623: end;
16624:
16625:
16626: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16627: begin
16628:  CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16629:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16630:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16631:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16632:    +'inationsRepeat, CombinationsRepeatDown )');
16633:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16634:   SIRegister_TComboSet(CL);
16635: end;
16636:
16637:  procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16638: begin
16639:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
16640:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )');
16641:   TProcOnReceiveCommand', 'Procedure (Sender: TObject; aCommand: Word; userParam : Integer)');
16642:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16643:    +'mmHandle : THandle; aString : String; userParam : Integer)');
16644:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16645:    +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer)');
16646:   SIRegister_TcyBaseComm(CL);
16647:  CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16648:  CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16649:  Function ValidateFileMappingName( aName : String) : String');
16650:  procedure makeCaption(leftSide, Rightside:string; form:TForm);
16651: end;
16652:
16653: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16654: begin
16655:   CL.AddTypeS('DERString', 'String');
16656:   CL.AddTypeS('DERChar', 'Char');
16657:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16658:    +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16659:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
16660:   CL.AddTypeS('DERNString', 'String');
16661:  const DERDecimalSeparator','String').SetString( '.');
16662:  const DERDefaultChars','String')('+°@/%-
      _.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16663:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz');
16664:  CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char) : Boolean');
16665:  Function isValidWebMailChar( aChar : Char) : Boolean');
16666:  Function isValidwebSite( aStr : String) : Boolean');
16667:  Function isValidWebMail( aStr : String) : Boolean');
16668:  Function ValidateDate( aDERStr : DERString; var RsltFormat : String) : Boolean');
16669:  Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16670:  Function IsDERChar( aChar : Char) : Boolean');
16671:  Function IsDERDefaultChar( aChar : Char) : Boolean');
16672:  Function IsDERMoneyChar( aChar : Char) : Boolean');
16673:  Function IsDERExceptionCar( aChar : Char) : Boolean');
16674:  Function IsDERSymbols( aDERString : String) : Boolean');
16675:  Function StringToDERCharSet( aStr : String) : DERString');
16676:  Function IsDERNDefaultChar( aChar : Char) : Boolean');
16677:  Function IsDERNChar( aChar : Char) : Boolean');
16678:  Function DERToDERNCharset( aDERStr : DERString) : DERNString');
16679:  Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean) : String');
```

```
16680:  Function DERExtractWebMail( aDERStr : DERString) : String');
16681:  Function DERExtractPhoneNr( aDERStr : DERString) : String');
16682:  Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
        SmartWebsiteRecognition:Boolean):TElementsType;
16683:  Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
        RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
        SmartWebsiteRecognition:Boolean):TElementsType;
16684:  Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
        aElementsType : TElementsType) : String;');
16685:  Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
        Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType);');
16686: end;
16687:
16688: procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16689: begin
16690:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16691:  Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16692:  Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16693:  Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16694:  Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16695:  Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16696:  Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
16697:  Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16698:  Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
        RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16699:  Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
        PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
        PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16700:  Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
        SingleDestinationColor, RefreshBmp : Boolean)');
16701:  Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
16702:  Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16703:  Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16704:  Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16705:  Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean);');
16706:  Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16707:  Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16708:  Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');
16709: end;
16710:
16711: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16712: begin
16713:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16714:   TPCMChannel', '( cMono, cStereo )');
16715:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16716:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16717:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
16718:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16719:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16720:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16721:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16722:    +'it48000Hz, Stereo16bit48000Hz )');
16723:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16724:    +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end');
16725:    tWaveFormatEx', 'PWaveFormatEx');
16726:    HMMIO', 'Integer');
16727:   TWaveDeviceFormats', 'set of TPCMFormat');
16728:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16729:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16730:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16731:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16732:   TWaveOutOptions', 'set of TWaveOutOption');
16733:   TStreamOwnership2', '( soReference, soOwned )');
16734:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16735:  // PRawWave', '^TRawWave // will not work');
16736:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16737:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioError');
16738:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError');
16739:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation');
16740:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16741:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16742:    +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16743:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16744:    +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16745:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16746:    +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16747:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16748:    +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16749:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16750:   TWaveAudioFilterEvent', 'Procedure (Sender : TObject; const Buffer:TObject; BufferSize:DWORD);
16751:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16752:  CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16753:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16754:  GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16755:  CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16756:  OpenStreamWaveAudio( Stream : TStream) : HMMIO');
16757:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16758:  GetAudioFormat( FormatTag : Word) : String');
16759:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) : String');
```

```
16760:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD');
16761:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD');
16762:  GetWaveAudioPeakLevel(const Data: TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16763:  InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16764:  SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16765:  ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16766:  MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
       TObject; BufferSize : DWORD) : Boolean');
16767:  ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
       dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16768:  SetPCMAudioFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
       TPCMSamplesPerSec; BitsPerSample : TPCMBitsPerSample)');
16769:  Procedure SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat)');
16770:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat');
16771:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD) : DWORD');
16772:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) : String');
16773:  WaitForSyncObject( SyncObject : THandle; Timeout : DWORD) : DWORD');
16774:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT');
16775: end;
16776:
16777: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);
16778: begin
16779:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16780:  CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16781:  'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
16782:  'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
16783:  'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
16784:  'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
16785:  'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
16786:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16787:   CL.AddTypeS('TPipeType', '( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg )');
16788:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16789:   SIRegister_TNamedPipe(CL);
16790:   SIRegister_TServerPipe(CL);
16791:   SIRegister_TClientPipe(CL);
16792:  CL.AddDelphiFunction('Function CalculateTimeout( aBasis : DWORD) : DWORD');
16793:  Function HasOverlappedIoCompleted( const ov : OVERLAPPED) : Boolean');
16794:  Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean):
       TOverlappedResult;
16795:  Function GetStreamAsText( stm : TStream) : string');
16796:  Procedure SetStreamAsText( const aTxt : string; stm : TStream)');
16797: end;
16798:
16799: procedure SIRegister_DPUtils(CL: TPSPascalCompiler);
16800: begin
16801:  // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16802:   SIRegister_TThumbData(CL);
16803:  'PIC_BMP','LongInt').SetInt( 0);
16804:  'PIC_JPG','LongInt').SetInt( 1);
16805:  'THUMB_WIDTH','LongInt').SetInt( 60);
16806:  'THUMB_HEIGHT','LongInt').SetInt( 60);
16807:  Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean');
16808:  Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)');
16809:  Function ReadBitmap( Filehandle : Integer; Width, Height : Integer) : TBitmap');
16810:  Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap)');
16811:  Function OpenPicture( fn : string; var tp : Integer) : Integer');
16812:  Function ConvertPicture( pi : Integer; tp : Integer) : TBitmap');
16813:  Function LoadPicture( fn : string; var w, h : Integer) : TBitmap');
16814:  Function TurnBitmap( bmp : TBitmap; ang : Integer) : TBitmap');
16815:  Function RotateBitmap( Bitmap : TBitmap; Direction : Integer) : TBitmap');
16816:  Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect');
16817:  Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer) : TBitmap');
16818:  Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer)');
16819:  Procedure FindFiles( path, mask : string; items : TStringList)');
16820:  Function LetFileName( s : string) : string');
16821:  Function LetParentPath( path : string) : string');
16822:  Function AddBackSlash( path : string) : string');
16823:  Function CutBackSlash( path : string) : string');
16824: end;
16825:
16826: procedure SIRegister_CommonTools(CL: TPSPascalCompiler);
16827: begin
16828:  //'BYTES','LongInt').SetInt( 1);
16829:  'KBYTES','LongInt').SetInt( 1024);
16830:  'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ));
16831:  'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ));
16832:  'DBG_GONE','LongWord').SetUInt( $99AC1D99);
16833:  'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D}');
16834:   SIRegister_MakeComServerMethodsPublic(CL);
16835:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16836:  Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) : Boolean');
16837:  Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean)');
16838:  Function TBGetTempFolder : string');
16839:  Function TBGetTempFile : string');
16840:  Function TBGetModuleFilename : string');
16841:  Function FormatModuleVersionInfo( const aFilename : string) : string');
16842:  Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD) : string');
16843:  Function TBGetFileSize( aFile : string; aMultipleOf : Integer) : Integer');
16844:  Function FormatAttribString( aAttr : Integer) : string');
```

```
16845:  Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo) : string');
16846:  Function ShellRecycle( aWnd : HWND; aFileOrFolder : string) : Boolean');
16847:  Function IsDebuggerPresent : BOOL');
16848:  Function TBNotImplemented : HRESULT');
16849: end;
16850:
16851: procedure SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
16852: begin
16853:   //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16854:   //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16855:   CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16856:   CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean;');
16857:   //TDrivesProperty = array['A'..'Z'] of boolean;
16858:  Function TBSetSystemTime( DateTime : TDateTime; DOW : word) : boolean');
16859:  Function IsElevated : Boolean');
16860:  Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
16861:   CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined )');
16862:  Function TrimNetResource( UNC : string) : string');
16863:  Procedure GetFreeDrives( var FreeDrives : TDrivesProperty)');
16864:  Procedure GetMappedDrives( var MappedDrives : TDrivesProperty)');
16865:  Function MapDrive( UNCPath : string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
       UserUsage : TPasswordUsage; User : string; Comment : string) : boolean');
16866:  Function UnmapDrive( Drive : char; Force : boolean) : boolean');
16867:  Function TBIsWindowsVista : Boolean');
16868:  Procedure SetVistaFonts( const AForm : TForm)');
16869:  Procedure SetVistaContentFonts( const AFont : TFont)');
16870:  Function GetProductType( var sType : String) : Boolean');
16871:  Function lstrcmp( lpString1, lpString2 : PChar) : Integer');
16872:  Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16873:  Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar');
16874:  Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
16875:  Function lstrcat( lpString1, lpString2 : PChar) : PChar');
16876:  Function lstrlen( lpString : PChar) : Integer');
16877:  Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16878:  Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD) : BOOL');
16879: end;
16880:
16881: procedure SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
16882: begin
16883:  'cLineTerminator','Char').SetString( #10);
16884:  'cLineTerminators','String').SetString( #13#10);
16885:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
16886:   SIRegister_TFixedCriticalSection(CL);
16887:   SIRegister_TMultiReadSingleWrite(CL);
16888:  CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16889:  Function GetDecimalSeparator : Char');
16890:  TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16891:  Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
       recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16892:   CL.AddTypeS('NativeInt', 'Integer');
16893:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16894:   CL.AddTypeS('NativeUInt', 'Cardinal');
16895:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16896:   //CL.AddTypeS('TBytes', 'array of Byte');
16897:   CL.AddTypeS('RawByteString', 'UnicodeString');
16898:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16899:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16900:   SIRegister_TPath(CL);
16901:   SIRegister_TFile(CL);
16902:   SIRegister_TdwsThread(CL);
16903:  Function GetSystemMilliseconds : Int64');
16904:  Function UTCDateTime : TDateTime');
16905:  Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString');
16906:  Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer');
16907:  Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer');
16908:  Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer');
16909:  Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer;');
16910:  Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer;');
16911:  Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString');
16912:  Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString');
16913:  Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer');
16914:  Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean');
16915:  Function InterlockedIncrement( var val : Integer) : Integer');
16916:  Function InterlockedDecrement( var val : Integer) : Integer');
16917:  Procedure FastInterlockedIncrement( var val : Integer)');
16918:  Procedure FastInterlockedDecrement( var val : Integer)');
16919:  Function InterlockedExchangePointer( var target : ___Pointer; val : ___Pointer) : ___Pointer');
16920:  Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
16921:  Procedure dwsOutputDebugString( const msg : UnicodeString)');
16922:  Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16923:  Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16924:  Procedure VarCopy( out dest : Variant; const src : Variant)');
16925:  Function VarToUnicodeStr( const v : Variant) : UnicodeString');
16926:  Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString');
16927:  Function LoadTextFromStream( aStream : TStream) : UnicodeString');
16928:  Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString');
16929:  Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
```

```
16930:  Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle');
16931:  Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle');
16932:  Procedure CloseFileHandle( hFile : THandle)');
16933:  Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean');
16934:  Function FileMove( const existing, new : UnicodeString) : Boolean');
16935:  Function dwsFileDelete( const fileName : String) : Boolean');
16936:  Function FileRename( const oldName, newName : String) : Boolean');
16937:  Function dwsFileSize( const name : String) : Int64');
16938:  Function dwsFileDateTime( const name : String) : TDateTime');
16939:  Function DirectSet8087CW( newValue : Word) : Word');
16940:  Function DirectSetMXCSR( newValue : Word) : Word');
16941:  Function TtoObject( const T: byte) : TObject');
16942:  Function TtoPointer( const T: byte) : ___Pointer');
16943:  Procedure GetMemForT(var T: byte; Size : integer)');
16944:  Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');
16945: end;
16946:
16947: procedure SIRegister_AdSocket(CL: TPSPascalCompiler);
16948: begin
16949:  'IPStrSize','LongInt').SetInt( 15);
16950:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
16951:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
16952:  'ADWSBASE','LongInt').SetInt( 9000);
16953:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
16954:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
16955:   SIRegister_EApdSocketException(CL);
16956:   TWsMode', '( wsClient, wsServer )');
16957:   TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket');
16958:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrCode : Integer)');
16959:   SIRegister_TApdSocket(CL);
16960: end;
16961:
16962: procedure SIRegister_AdPort(CL: TPSPascalCompiler);
16963: begin
16964:   SIRegister_TApdCustomComPort(CL);
16965:   SIRegister_TApdComPort(CL);
16966:  Function ComName( const ComNumber : Word) : ShortString');
16967:  Function SearchComPort( const C : TComponent) : TApdCustomComPort');
16968: end;
16969:
16970: procedure SIRegister_PathFunc(CL: TPSPascalCompiler);
16971: begin
16972:  Function inAddBackslash( const S : String) : String');
16973:  Function PathChangeExt( const Filename, Extension : String) : String');
16974:  Function PathCharCompare( const S1, S2 : PChar) : Boolean');
16975:  Function PathCharIsSlash( const C : Char) : Boolean');
16976:  Function PathCharIsTrailByte( const S : String; const Index : Integer) : Boolean');
16977:  Function PathCharLength( const S : String; const Index : Integer) : Integer');
16978:  Function inPathCombine( const Dir, Filename : String) : String');
16979:  Function PathCompare( const S1, S2 : String) : Integer');
16980:  Function PathDrivePartLength( const Filename : String) : Integer');
16981:  Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;
16982:  Function inPathExpand( const Filename : String) : String');
16983:  Function PathExtensionPos( const Filename : String) : Integer');
16984:  Function PathExtractDir( const Filename : String) : String');
16985:  Function PathExtractDrive( const Filename : String) : String');
16986:  Function PathExtractExt( const Filename : String) : String');
16987:  Function PathExtractName( const Filename : String) : String');
16988:  Function PathExtractPath( const Filename : String) : String');
16989:  Function PathIsRooted( const Filename : String) : Boolean');
16990:  Function PathLastChar( const S : String) : PChar');
16991:  Function PathLastDelimiter( const Delimiters, S : string) : Integer');
16992:  Function PathLowercase( const S : String) : String');
16993:  Function PathNormalizeSlashes( const S : String) : String');
16994:  Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
16995:  Function PathPos( Ch : Char; const S : String) : Integer');
16996:  Function PathStartsWith( const S, AStartsWith : String) : Boolean');
16997:  Function PathStrNextChar( const S : PChar) : PChar');
16998:  Function PathStrPrevChar( const Start, Current : PChar) : PChar');
16999:  Function PathStrScan( const S : PChar; const C : Char) : PChar');
17000:  Function inRemoveBackslash( const S : String) : String');
17001:  Function RemoveBackslashUnlessRoot( const S : String) : String');
17002:  Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean)');
17003: end;
17004:
17005:
17006: procedure SIRegister_CmnFunc2(CL: TPSPascalCompiler);
17007: begin
17008:  NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17009:  NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17010:  NEWREGSTR_PATH_SPECIAL_FOLDERS','String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17011:  NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17012:  NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName');
17013:  NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString');
17014:  KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
17015:   //CL.AddTypeS('PLeadByteSet', '^TLeadByteSet // will not work');
17016:   CL.AddTypeS('TLeadByteSet', 'set of Char');
17017:   SIRegister_TOneShotTimer(CL);
17018:   CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
```

```
17019:  'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17020:  Function NewFileExists( const Name : String) : Boolean');
17021:  Function inDirExists( const Name : String) : Boolean');
17022:  Function FileOrDirExists( const Name : String) : Boolean');
17023:  Function IsDirectoryAndNotReparsePoint( const Name : String) : Boolean');
17024:  Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String');
17025:  Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17026:  Function GetIniBool( const Section,Key:String; const Default:Boolean;const Filename:String): Boolean');
17027:  Function IniKeyExists( const Section, Key, Filename : String) : Boolean');
17028:  Function IsIniSectionEmpty( const Section, Filename : String) : Boolean');
17029:  Function SetIniString( const Section, Key, Value, Filename : String) : Boolean');
17030:  Function SetIniInt(const Section,Key:String;const Value: Longint;const Filename: String):Boolean');
17031:  Function SetIniBool(const Section,Key:String;const Value: Boolean;const Filename: String):Boolean');
17032:  Procedure DeleteIniEntry( const Section, Key, Filename : String)');
17033:  Procedure DeleteIniSection( const Section, Filename : String)');
17034:  Function GetEnv( const EnvVar : String) : String');
17035:  Function GetCmdTail : String');
17036:  Function GetCmdTailEx( StartIndex : Integer) : String');
17037:  Function NewParamCount : Integer');
17038:  Function NewParamStr( Index : Integer) : string');
17039:  Function AddQuotes( const S : String) : String');
17040:  Function RemoveQuotes( const S : String) : String');
17041:  Function inGetShortName( const LongName : String) : String');
17042:  Function inGetWinDir : String');
17043:  Function inGetSystemDir : String');
17044:  Function GetSysWow64Dir : String');
17045:  Function GetSysNativeDir( const IsWin64 : Boolean) : String');
17046:  Function inGetTempDir : String');
17047:  Function StringChange( var S : String; const FromStr, ToStr : String) : Integer');
17048:  Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17049:  Function AdjustLength( var S : String; const Res : Cardinal) : Boolean');
17050:  Function UsingWinNT : Boolean');
17051:  Function ConvertConstPercentStr( var S : String) : Boolean');
17052:  Function ConvertPercentStr( var S : String) : Boolean');
17053:  Function ConstPos( const Ch : Char; const S : String) : Integer');
17054:  Function SkipPastConst( const S : String; const Start : Integer) : Integer');
17055:  Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String) : Boolean');
17056:  Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String):Boolean;
17057:  Function RegValueExists( H : HKEY; Name : PChar) : Boolean');
17058:  Function RegCreateKeyExView(const RegView:TRegView;hKey:HKEY;lpSubKey:PChar;Reserved:DWORD;lpClass:PChar;
        dwOptions:DWORD;samDesired:REGSAM;lpSecurityAttributes:TObject;var
        phkResult:HKEY;lpdwDisposition:DWORD):Longint;
17059:  Function RegOpenKeyExView(const
        RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17060:  Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar) : Longint;
17061:  Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17062:  Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyName:PChar):Longint;
17063:  Function GetShellFolderPath( const FolderID : Integer) : String');
17064:  Function IsAdminLoggedOn : Boolean');
17065:  Function IsPowerUserLoggedOn : Boolean');
17066:  Function IsMultiByteString( const S : AnsiString) : Boolean');
17067:  Function FontExists( const FaceName : String) : Boolean');
17068:  //Procedure FreeAndNil( var Obj)');
17069:  Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT) : HMODULE');
17070:  Function GetUILanguage : LANGID');
17071:  Function RemoveAccelChar( const S : String) : String');
17072:  Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17073:  Function AddPeriod( const S : String) : String');
17074:  Function GetExceptMessage : String');
17075:  Function GetPreferredUIFont : String');
17076:  Function IsWildcard( const Pattern : String) : Boolean');
17077:  Function WildcardMatch( const Text, Pattern : PChar) : Boolean');
17078:  Function IntMax( const A, B : Integer) : Integer');
17079:  Function Win32ErrorString( ErrorCode : Integer) : String');
17080:  Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet)');
17081:  Function inCompareMem( P1, P2 : TObject; Length : Integer) : Boolean');
17082:  Function DeleteDirTree( const Dir : String) : Boolean');
17083:  Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
17084:  Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT)');
17085:  // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17086:  Function inCharInSet( C : Char; const CharSet : TSysCharSet) : Boolean');
17087:  Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String) : Boolean');
17088:  Function ShutdownBlockReasonDestroy( Wnd : HWND) : Boolean');
17089:  Function TryStrToBoolean( const S : String; var BoolResult : Boolean) : Boolean');
17090:  Procedure WaitMessageWithTimeout( const Milliseconds : DWORD)');
17091:  Function MoveFileReplace(const ExistingFileName, NewFileName : String) : Boolean');
17092:  Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND)');
17093: end;
17094:
17095: procedure SIRegister_CmnFunc(CL: TPSPascalCompiler);
17096: begin
17097:   SIRegister_TWindowDisabler(CL);
17098:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17099:   TMsgBoxCallbackFunc','procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17100:  Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox)');
17101:  Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17102:  Function AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Integer');
17103:  Function MsgBoxP( const Text,Caption: PChar; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17104:  Function inMsgBox(const Text,Caption:String; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
```

```
17105:  Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
       Typ:TMsgBoxType;const Buttons:Cardinal):Integer');
17106:  Procedure ReactivateTopWindow');
17107:  Procedure SetMessageBoxCaption( const Typ : TMsgBoxType; const NewCaption : PChar)');
17108:  Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean)');
17109:  Procedure SetMessageBoxCallbackFunc(const AFunc : TMsgBoxCallbackFunc; const AParam : LongInt)');
17110: end;
17111:
17112: procedure SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17113: begin
17114:   SIRegister_TImageGrabber(CL);
17115:   SIRegister_TCaptureDrivers(CL);
17116:   SIRegister_TCaptureDriver(CL);
17117: end;
17118:
17119: procedure SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17120: begin
17121:  Function GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:String;const
       Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean');
17122:  Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
       Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean');
17123: end;
17124:
17125: procedure SIRegister_RedirFunc(CL: TPSPascalCompiler);
17126: begin
17127:   CL.AddTypeS('TPreviousFsRedirectionState', 'record DidDisable : Boolean; OldValue : ___Pointer; end');
17128:  CL.AddDelphiFunction('Function AreFsRedirectionFunctionsAvailable : Boolean');
17129:  Function DisableFsRedirectionIf(const Disable:Boolean;var PreviousState:TPreviousFsRedirectionState):Boo;
17130:  Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState)');
17131:  Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17132:  Function CreateProcessRedir(const DisableFsRedir:Boolean; const lpApplicationName : PChar; const
       lpCommandLine : PChar; const lpProcessAttributes, lpThreadAttributes : PSecurityAttributes; const
       bInheritHandles : BOOL;'
17133:  +' const dwCreationFlags : DWORD; const lpEnvironment : Pointer; const lpCurrentDirectory : PChar; const
       lpStartupInfo : TStartupInfo; var lpProcessInformation : TProcessInformation) : BOOL');
17134:  Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
       FailIfExists : BOOL) : BOOL');
17135:  Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17136:  Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17137:  Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17138:  Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
       TWin32FindData) : THandle');
17139:  Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String) : DWORD');
17140:  Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String) : String');
17141:  Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
       TFileVersionNumbers) : Boolean');
17142:  Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17143:  Function MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
17144:  Function MoveFileExRedir(const DisableFsRedir:Boolean;const ExistingFilename,NewFilename:String;const
       Flags:DWORD):BOOL;
17145:  Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17146:  Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17147:  Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17148:  Function SetNTFSCompressionRedir(const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17149:   SIRegister_TFileRedir(CL);
17150:   SIRegister_TTextFileReaderRedir(CL);
17151:   SIRegister_TTextFileWriterRedir(CL);
17152: end;
17153:
17154: procedure SIRegister_Int64Em(CL: TPSPascalCompiler);
17155: begin
17156:   //CL.AddTypeS('LongWord', 'Cardinal');
17157:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17158:  Function Compare64( const N1, N2 : Integer64) : Integer');
17159:  Procedure Dec64( var X : Integer64; N : LongWord)');
17160:  Procedure Dec6464( var X : Integer64; const N : Integer64)');
17161:  Function Div64( var X : Integer64; const Divisor : LongWord) : LongWord');
17162:  Function Inc64( var X : Integer64; N : LongWord) : Boolean');
17163:  Function Inc6464( var X : Integer64; const N : Integer64) : Boolean');
17164:  Function Integer64ToStr( X : Integer64) : String');
17165:  Function Mod64( const X : Integer64; const Divisor : LongWord) : LongWord');
17166:  Function Mul64( var X : Integer64; N : LongWord) : Boolean');
17167:  Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64)');
17168:  Procedure Shr64( var X : Integer64; Count : LongWord)');
17169:  Function StrToInteger64( const S : String; var X : Integer64) : Boolean');
17170: end;
17171:
17172: procedure SIRegister_InstFunc(CL: TPSPascalCompiler);
17173: begin
17174:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17175:   SIRegister_TSimpleStringList(CL);
17176:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle)');
17177:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17178:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17179:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17180:  // TMD5Digest = array[0..15] of Byte;
17181:   //     TSHA1Digest = array[0..19] of Byte;
17182:  Function CheckForMutexes( Mutexes : String) : Boolean');
17183:  Function CreateTempDir : String');
```

```
17184:  Function DecrementSharedCount( const RegView : TRegView; const Filename : String) : Boolean');
17185:  Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
        FirstRetryDelayMS, SubsequentRetryDelayMS : Integer)');
17186:  //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
        DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
        TDeleteFileProc; const Param : Pointer) : Boolean');
17187:  //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
        TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer) :
        TDetermineDefaultLanguageResult');
17188:  //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17189:  Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String) : Boolean');
17190:  Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17191:  Function GetComputerNameString : String');
17192:  Function GetFileDateTime(const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime):Bool;
17193:  Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String) : TMD5Digest');
17194:  Function GetMD5OfAnsiString( const S : AnsiString) : TMD5Digest');
17195: // Function GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest');
17196:  Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest');
17197:  Function GetSHA1OfAnsiString( const S : AnsiString) : TSHA1Digest');
17198: // Function GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest');
17199:  Function GetRegRootKeyName( const RootKey : HKEY) : String');
17200:  Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
        TotBytes:Int64):Bool;
17201:  Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
        TotalBytes: Integer64):Bool;
17202:  Function GetUserNameString : String');
17203:  Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17204:  //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
        String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
        ResultCode:Integer) : Boolean');
17205: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
        TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17206:  Procedure InternalError( const Id : String)');
17207:  Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17208:  Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean');
17209:  Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17210:  Function MakePendingFileRenameOperationsChecksum : TMD5Digest');
17211:  Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17212:  Procedure RaiseFunctionFailedError( const FunctionName : String)');
17213:  Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17214:  Procedure RefreshEnvironment');
17215:  Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17216:  Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17217:  Procedure UnregisterFont( const FontName, FontFilename : String)');
17218:  Function RestartComputer : Boolean');
17219:  Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17220:  Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String)');
17221:  Procedure Win32ErrorMsg( const FunctionName : String)');
17222:  Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17223:  Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17224:  //from Func2
17225:  //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
        IconFilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean; '
17226:    //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String');
17227:  Procedure RegisterTypeLibrary( const Filename : String)');
17228:  //Procedure UnregisterTypeLibrary( const Filename : String)');
17229:  //Function UnpinShellLink( const Filename : String) : Boolean');}
17230:  function getVersionInfoEx3: TOSVersionInfoEx;');
17231:  Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17232:  procedure InitOle;');
17233:  Function ExpandConst( const S : String) : String');
17234:  Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17235:  Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17236:  Function ExpandConstIfPrefixed( const S : String) : String');
17237:  Procedure LogWindowsVersion');
17238:  Function EvalCheck( const Expression : String) : Boolean');
17239: end;
17240:
17241: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17242: begin
17243:   CL.AddClassN(CL.FindClass('TOBJECT'),'TResourceDetails');
17244:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17245:   SIRegister_TResourceModule(CL);
17246:   SIRegister_TResourceDetails(CL);
17247:   SIRegister_TAnsiResourceDetails(CL);
17248:   SIRegister_TUnicodeResourceDetails(CL);
17249:  Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17250:  Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17251:  Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17252:  Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17253:  Function ResourceNameToInt( const s : string) : Integer');
17254:  Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer');
17255: end;
17256:
17257:
17258: procedure SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17259: begin
17260:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17261:   with CL.AddClassN(CL.FindClass('TObject'),'TSimpleComPort') do begin
```

```
17262:     RegisterMethod('Constructor Create');
17263:     RegisterMethod('Procedure Free');
17264:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17265:     RegisterMethod('Procedure WriteString( const S : String)');
17266:     RegisterMethod('Procedure ReadString( var S : String)');
17267:   end;
17268:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17269:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17270:       SimpleComPort.WriteString(AsciiChar);
17271: end;
17272:
17273:
17274: procedure SIRegister_Console(CL: TPSPascalCompiler);
17275: begin
17276:  CL.AddConstantN('White','LongInt').SetInt( 15); }
17277: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17278:  CL.AddConstantN('conBW40','LongInt').SetInt( 0);
17279:  CL.AddConstantN('conCO40','LongInt').SetInt( 1);
17280:  CL.AddConstantN('conBW80','LongInt').SetInt( 2);
17281:  CL.AddConstantN('conCO80','LongInt').SetInt( 3);
17282:  CL.AddConstantN('conMono','LongInt').SetInt( 7);
17283:  CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17284:  //CL.AddConstantN('C40','').SetString( CO40);
17285:  //CL.AddConstantN('C80','').SetString( CO80);
17286:  Function conReadKey : Char');
17287:  Function conKeyPressed : Boolean');
17288:  Procedure conGotoXY( X, Y : Smallint)');
17289:  Function conWhereX : Integer');
17290:  Function conWhereY : Integer');
17291:  Procedure conTextColor( Color : Byte);');
17292:  Function conTextColor1 : Byte;');
17293:  Procedure conTextBackground( Color : Byte);');
17294:  Function conTextBackground1 : Byte;');
17295:  Procedure conTextMode( Mode : Word)');
17296:  Procedure conLowVideo');
17297:  Procedure conHighVideo');
17298:  Procedure conNormVideo');
17299:  Procedure conClrScr');
17300:  Procedure conClrEol');
17301:  Procedure conInsLine');
17302:  Procedure conDelLine');
17303:  Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17304:  Function conScreenWidth : Smallint');
17305:  Function conScreenHeight : Smallint');
17306:  Function conBufferWidth : Smallint');
17307:  Function conBufferHeight : Smallint');
17308:  procedure InitScreenMode;');
17309: end;
17310:
17311:
17312: {A simple Oscilloscope using  TWaveIn class.
17313:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
17314: uses
17315:   Forms,
17316:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
17317:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
17318:   uColorFunctions in 'uColorFunctions.pas',
17319:   AMixer in 'AMixer.pas',
17320:   uSettings in 'uSettings.pas',
17321:   UWavein4 in 'UWavein4.pas',
17322:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
17323:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
17324:
17325:
17326: Functions_max hex in the box maXbox
17327: functionslist.txt
17328: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98
17329:
17330: ********************************************************************************
17331: Procedure
17332: PROCEDURE SIZE 8073 7507 7401 6792 6310 5971 4438 3797 3600 7881 7938
17333: Procedure ***************************Now the Procedure list*******************
17334: Procedure ( ACol, ARow : Integer; Items : TStrings)
17335: Procedure ( Agg : TAggregate)
17336: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
17337: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
17338: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
17339: Procedure ( ASender : TObject; const ABytes : Integer)
17340: Procedure ( ASender : TObject; VStream : TStream)
17341: Procedure ( AThread : TIdThread)
17342: Procedure ( AWebModule : TComponent)
17343: Procedure ( Column : TColumn)
17344: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
17345: Procedure ( const iStart : integer; const sText : string)
17346: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
17347: Procedure ( Database : TDatabase; LoginParams : TStrings)
17348: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
17349: Procedure ( DATASET : TDATASET)
```

```
17350: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
17351: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
17352: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
17353: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
17354: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
17355: Procedure ( Done : Integer)
17356: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
17357: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
17358: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
17359: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
17360: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
17361: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
17362: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
17363: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
17364: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
17365: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
17366: Procedure ( SENDER : TFIELD; const TEXT : String)
17367: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
17368: Procedure ( Sender : TIdTelnet; const Buffer : String)
17369: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
17370: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
17371: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17372: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
17373: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
17374: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
17375: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
17376: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
17377: Procedure ( Sender : TObject; Button : TMPBtnType)
17378: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
17379: Procedure ( Sender : TObject; Button : TUDBtnType)
17380: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
17381: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
17382: Procedure ( Sender : TObject; Column : TListColumn)
17383: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
17384: Procedure ( Sender : TObject; Connecting : Boolean)
17385: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
17386: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
17387: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
17388: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
17389: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
17390: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
17391: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
17392: Procedure ( Sender : TObject; Index : LongInt)
17393: Procedure ( Sender : TObject; Item : TListItem)
17394: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
17395: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
17396: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
17397: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
17398: Procedure ( Sender : TObject; Item : TListItem; var S : string)
17399: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
17400: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
17401: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
17402: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
17403: Procedure ( Sender : TObject; Node : TTreeNode)
17404: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
17405: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
17406: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
17407: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
17408: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
17409: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
17410: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
17411: Procedure ( Sender : TObject; Rect : TRect)
17412: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
17413: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
17414: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
17415: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
17416: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
17417: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
17418: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
17419: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
17420: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
17421: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
17422: Procedure ( Sender : TObject; Thread : TServerClientThread)
17423: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
17424: Procedure ( Sender : TObject; Username, Password : string)
17425: Procedure ( Sender : TObject; var AllowChange : Boolean)
17426: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
17427: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
17428: Procedure ( Sender : TObject; var Continue : Boolean)
17429: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
17430: Procedure ( Sender : TObject; var Username : string)
17431: Procedure ( Sender : TObject; Wnd : HWND)
17432: Procedure ( Sender : TToolbar; Button : TToolButton)
17433: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
17434: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
17435: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
17436: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
17437: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
```

```
17438: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
17439: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
17440: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
17441: procedure (Sender: TObject)
17442: procedure (Sender: TObject; var Done: Boolean)
17443: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
17444: procedure _T(Name: tbtString; v: Variant);
17445: Procedure AbandonSignalHandler( RtlSigNum : Integer)
17446: Procedure Abort
17447: Procedure About1Click( Sender : TObject)
17448: Procedure Accept( Socket : TSocket)
17449: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
17450: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
17451: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
17452: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
17453: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
17454: Procedure Add( Addend1, Addend2 : TMyBigInt)
17455: Procedure ADD( const AKEY, AVALUE : VARIANT)
17456: Procedure Add( const Key : string; Value : Integer)
17457: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
17458: Procedure ADD( FIELD : TFIELD)
17459: Procedure ADD( ITEM : TMENUITEM)
17460: Procedure ADD( POPUP : TPOPUPMENU)
17461: Procedure AddCharacters( xCharacters : TCharSet)
17462: Procedure AddDriver( const Name : string; List : TStrings)
17463: Procedure AddImages( Value : TCustomImageList)
17464: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
17465: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
17466: Procedure AddLoader( Loader : TBitmapLoader)
17467: Procedure ADDPARAM( VALUE : TPARAM)
17468: Procedure AddPassword( const Password : string)
17469: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
17470: Procedure AddState( oState : TniRegularExpressionState)
17471: Procedure AddStrings( Strings : TStrings);
17472: procedure AddStrings(Strings: TStrings);
17473: Procedure AddStrings1( Strings : TWideStrings);
17474: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
17475: Procedure AddToRecentDocs( const Filename : string)
17476: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
17477: Procedure AllFunctionsList1Click( Sender : TObject)
17478: procedure AllObjectsList1Click(Sender: TObject);
17479: Procedure Allocate( AAllocateBytes : Integer)
17480: procedure AllResourceList1Click(Sender: TObject);
17481: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
17482: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
17483: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
17484: Procedure AnsiFree( var s : AnsiString)
17485: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
17486: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
17487: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
17488: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
17489: Procedure AntiFreeze;
17490: Procedure APPEND
17491: Procedure Append( const S : WideString)
17492: procedure Append(S: string);
17493: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
17494: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
17495: Procedure AppendChunk( Val : OleVariant)
17496: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
17497: Procedure AppendStr( var Dest : string; S : string)
17498: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
17499: Procedure ApplyRange
17500: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17501: Procedure Arrange( Code : TListArrangement)
17502: procedure Assert(expr : Boolean; const msg: string);
17503: procedure Assert2(expr : Boolean; const msg: string);
17504: Procedure Assign( AList : TCustomBucketList)
17505: Procedure Assign( Other : TObject)
17506: Procedure Assign( Source : TDragObject)
17507: Procedure Assign( Source : TPersistent)
17508: Procedure Assign(Source: TPersistent)
17509: procedure Assign2(mystring, mypath: string);
17510: Procedure AssignCurValues( Source : TDataSet);
17511: Procedure AssignCurValues1( const CurValues : Variant);
17512: Procedure ASSIGNFIELD( FIELD : TFIELD)
17513: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
17514: Procedure AssignFile(var F: Text; FileName: string)
17515: procedure AssignFile(var F: TextFile; FileName: string)
17516: procedure AssignFileRead(var mystring, myfilename: string);
17517: procedure AssignFileWrite(mystring, myfilename: string);
17518: Procedure AssignTo( Other : TObject)
17519: Procedure AssignValues( Value : TParameters)
17520: Procedure ASSIGNVALUES( VALUE : TPARAMS)
17521: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
17522: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
17523: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
17524: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
17525: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17526: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
```

```
17527: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
17528: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
17529: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
17530: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
17531: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
17532: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
17533: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
17534: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17535: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
17536: procedure Beep
17537: Procedure BeepOk
17538: Procedure BeepQuestion
17539: Procedure BeepHand
17540: Procedure BeepExclamation
17541: Procedure BeepAsterisk
17542: Procedure BeepInformation
17543: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
17544: Procedure BeginLayout
17545: Procedure BeginTimer( const Delay, Resolution : Cardinal)
17546: Procedure BeginUpdate
17547: procedure BeginUpdate;
17548: procedure BigScreen1Click(Sender: TObject);
17549: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
17550: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
17551: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
17552: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
17553: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
17554: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
17555: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
17556: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
17557: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
17558: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
17559: Procedure BreakPointMenuClick( Sender : TObject)
17560: procedure BRINGTOFRONT
17561: procedure BringToFront;
17562: Procedure btnBackClick( Sender : TObject)
17563: Procedure btnBrowseClick( Sender : TObject)
17564: Procedure BtnClick( Index : TNavigateBtn)
17565: Procedure btnLargeIconsClick( Sender : TObject)
17566: Procedure BuildAndSendRequest( AURI : TIdURI)
17567: Procedure BuildCache
17568: Procedure BurnMemory( var Buff, BuffLen : integer)
17569: Procedure BurnMemoryStream( Destructo : TMemoryStream)
17570: Procedure CalculateFirstSet
17571: Procedure Cancel
17572: procedure CancelDrag;
17573: Procedure CancelEdit
17574: procedure CANCELHINT
17575: Procedure CancelRange
17576: Procedure CancelUpdates
17577: Procedure CancelWriteBuffer
17578: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
17579: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
17580: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
17581: procedure CaptureScreenFormat(vname: string; vextension: string);
17582: procedure CaptureScreenPNG(vname: string);
17583: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
17584: procedure CASCADE
17585: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
17586: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
17587: Procedure cbPathClick( Sender : TObject)
17588: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
17589: Procedure cedebugAfterExecute( Sender : TPSScript)
17590: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
17591: Procedure cedebugCompile( Sender : TPSScript)
17592: Procedure cedebugExecute( Sender : TPSScript)
17593: Procedure cedebugIdle( Sender : TObject)
17594: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
17595: Procedure CenterHeight( const pc, pcParent : TControl)
17596: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
17597: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
17598: Procedure Change
17599: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
17600: Procedure Changed
17601: Procedure ChangeDir( const ADirName : string)
17602: Procedure ChangeDirUp
17603: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
17604: Procedure ChangeLevelBy( Value : TChangeRange)
17605: Procedure ChDir(const s: string)
17606: Procedure Check(Status: Integer)
17607: Procedure CheckCommonControl( CC : Integer)
17608: Procedure CHECKFIELDNAME( const FIELDNAME : String)
17609: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
17610: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
17611: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
17612: Procedure CheckToken( T : Char)
17613: procedure CheckToken(t:char)
17614: Procedure CheckTokenSymbol( const S : string)
17615: procedure CheckTokenSymbol(s:string)
```

```
17616: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
17617: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17618: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
17619: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
17620: procedure CipherFile1Click(Sender: TObject);
17621: Procedure Clear;
17622: Procedure Clear1Click( Sender : TObject)
17623: Procedure ClearColor( Color : TColor)
17624: Procedure CLEARITEM( AITEM : TMENUITEM)
17625: Procedure ClearMapping
17626: Procedure ClearSelection( KeepPrimary : Boolean)
17627: Procedure ClearWriteBuffer
17628: Procedure Click
17629: Procedure Close
17630: Procedure Close1Click( Sender : TObject)
17631: Procedure CloseDatabase( Database : TDatabase)
17632: Procedure CloseDataSets
17633: Procedure CloseDialog
17634: Procedure CloseFile(var F: Text);
17635: Procedure Closure
17636: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17637: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17638: Procedure CodeCompletionList1Click( Sender : TObject)
17639: Procedure ColEnter
17640: Procedure Collapse
17641: Procedure Collapse( Recurse : Boolean)
17642: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
17643: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
17644: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
17645: Procedure Compile1Click( Sender : TObject)
17646: procedure ComponentCount1Click(Sender: TObject);
17647: Procedure Compress(azipfolder, azipfile: string)
17648: Procedure DeCompress(azipfolder, azipfile: string)
17649: Procedure XZip(azipfolder, azipfile: string)
17650: Procedure XUnZip(azipfolder, azipfile: string)
17651: Procedure Connect( const ATimeout : Integer)
17652: Procedure Connect( Socket : TSocket)
17653: procedure Console1Click(Sender: TObject);
17654: Procedure Continue
17655: Procedure ContinueCount( var Counter : TJclCounter)
17656: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
17657: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
17658: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
17659: Procedure ConvertImage(vsource, vdestination: string);
17660: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
17661: Procedure ConvertBitmap(vsource, vdestination: string);
17662: Procedure ConvertToGray(Cnv: TCanvas);
17663: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
17664: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
17665: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
17666: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
17667: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
17668: Procedure CopyFrom( mbCopy : TMyBigInt)
17669: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
17670: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
17671: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
17672: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
17673: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
17674: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
17675: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
17676: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
17677: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
17678: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17679: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
17680: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17681: Procedure CopyToClipboard
17682: Procedure CountParts
17683: Procedure CreateDataSet
17684: Procedure CreateEmptyFile( const FileName : string)
17685: Procedure CreateFileFromString( const FileName, Data : string)
17686: Procedure CreateFromDelta( Source : TPacketDataSet)
17687: procedure CREATEHANDLE
17688: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
17689: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
17690: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
17691: Procedure CreateTable
17692: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
17693: procedure CSyntax1Click(Sender: TObject);
17694: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
17695: Procedure CURSORPOSCHANGED
17696: procedure CutFirstDirectory(var S: String)
17697: Procedure DataBaseError(const Message: string)
17698: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
17699: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
17700: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
17701: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
```

```
17702: Procedure DBIError(errorCode: Integer)
17703: Procedure DebugOutput( const AText : string)
17704: Procedure DebugRun1Click( Sender : TObject)
17705: procedure Dec;
17706: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
17707: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
17708: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
17709: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
17710: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
17711: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
17712: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
17713: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
17714: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
17715: Procedure Decompile1Click( Sender : TObject)
17716: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
17717: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
17718: Procedure DeferLayout
17719: Procedure defFileread
17720: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
17721: Procedure DelayMicroseconds( const MicroSeconds : Integer)
17722: Procedure Delete
17723: Procedure Delete( const AFilename : string)
17724: Procedure Delete( const Index : Integer)
17725: Procedure DELETE( INDEX : INTEGER)
17726: Procedure Delete( Index : LongInt)
17727: Procedure Delete( Node : TTreeNode)
17728: procedure Delete(var s: AnyString; ifrom, icount: Longint);
17729: Procedure DeleteAlias( const Name : string)
17730: Procedure DeleteDriver( const Name : string)
17731: Procedure DeleteIndex( const Name : string)
17732: Procedure DeleteKey( const Section, Ident : String)
17733: Procedure DeleteRecords
17734: Procedure DeleteRecords( AffectRecords : TAffectRecords)
17735: Procedure DeleteString( var pStr : String; const pDelStr : string)
17736: Procedure DeleteTable
17737: procedure DelphiSite1Click(Sender: TObject);
17738: Procedure Deselect
17739: Procedure Deselect( Node : TTreeNode)
17740: procedure DestroyComponents
17741: Procedure DestroyHandle
17742: Procedure Diff( var X : array of Double)
17743: procedure Diff(var X: array of Double);
17744: Procedure DirCreate( const DirectoryName : String)');
17745: procedure DISABLEALIGN
17746: Procedure DisableConstraints
17747: Procedure Disconnect
17748: Procedure Disconnect( Socket : TSocket)
17749: Procedure Dispose
17750: procedure Dispose(P: PChar)
17751: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
17752: Procedure DoKey( Key : TDBCtrlGridKey)
17753: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17754: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17755: Procedure Dormant
17756: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
17757: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
17758: Procedure DoubleToComp( Value : Double; var Result : Comp)
17759: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
17760: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
17761: procedure Draw(X, Y: Integer; Graphic: TGraphic);
17762: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
17763: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17764: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
17765: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17766: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
17767: procedure DrawFocusRect(const Rect: TRect);
17768: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
17769: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17770: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
17771: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
17772: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
17773: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
17774: Procedure DropConnections
17775: Procedure DropDown
17776: Procedure DumpDescription( oStrings : TStrings)
17777: Procedure DumpStateTable( oStrings : TStrings)
17778: Procedure EDIT
17779: Procedure EditButtonClick
17780: Procedure EditFont1Click( Sender : TObject)
17781: procedure Ellipse(X1, Y1, X2, Y2: Integer);
17782: Procedure Ellipse1( const Rect : TRect);
17783: Procedure EMMS
17784: Procedure Encode( ADest : TStream)
17785: procedure ENDDRAG(DROP:BOOLEAN)
17786: Procedure EndEdit( Cancel : Boolean)
17787: Procedure EndTimer
17788: Procedure EndUpdate
```

```
17789: Procedure EraseSection( const Section : string)
17790: Procedure Error( const Ident : string)
17791: procedure Error(Ident:Integer)
17792: Procedure ErrorFmt( const Ident : string; const Args : array of const)
17793: Procedure ErrorStr( const Message : string)
17794: procedure ErrorStr(Message:String)
17795: Procedure Exchange( Index1, Index2 : Integer)
17796: procedure Exchange(Index1, Index2: Integer);
17797: Procedure Exec( FileName, Parameters, Directory : string)
17798: Procedure ExecProc
17799: Procedure ExecSQL( UpdateKind : TUpdateKind)
17800: Procedure Execute
17801: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
17802: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
17803: Procedure ExecuteCommand(executeFile, paramstring: string)
17804: Procedure ExecuteShell(executeFile, paramstring: string)
17805: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
17806: Procedure ExitThread(ExitCode: Integer); stdcall;
17807: Procedure ExitProcess(ExitCode: Integer); stdcall;
17808: Procedure Expand( AUserName : String; AResults : TStrings)
17809: Procedure Expand( Recurse : Boolean)
17810: Procedure ExportClipboard1Click( Sender : TObject)
17811: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
17812: Procedure ExtractContentFields( Strings : TStrings)
17813: Procedure ExtractCookieFields( Strings : TStrings)
17814: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
17815: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
17816: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
17817: Procedure ExtractQueryFields( Strings : TStrings)
17818: Procedure FastDegToGrad
17819: Procedure FastDegToRad
17820: Procedure FastGradToDeg
17821: Procedure FastGradToRad
17822: Procedure FastRadToDeg
17823: Procedure FastRadToGrad
17824: Procedure FileClose( Handle : Integer)
17825: Procedure FileClose(handle: integer)
17826: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
17827: Procedure FileStructure( AStructure : TIdFTPDataStructure)
17828: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
17829: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
17830: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
17831: Procedure FillChar2(var X: PChar ; count: integer; value: char)
17832: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
17833: Procedure FillIPList
17834: procedure FillRect(const Rect: TRect);
17835: Procedure FillTStrings( AStrings : TStrings)
17836: Procedure FilterOnBookmarks( Bookmarks : array of const)
17837: procedure FinalizePackage(Module: HMODULE)
17838: procedure FindClose;
17839: procedure FindClose2(var F: TSearchRec)
17840: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
17841: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
17842: Procedure FindNearest( const KeyValues : array of const)
17843: Procedure FinishContext
17844: Procedure FIRST
17845: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
17846: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
17847: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
17848: Procedure FlushSchemaCache( const TableName : string)
17849: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
17850: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
17851: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
17852: Procedure FormActivate( Sender : TObject)
17853: procedure FormatLn(const format: String; const args: array of const);  //alias
17854: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
17855: Procedure FormCreate( Sender : TObject)
17856: Procedure FormDestroy( Sender : TObject)
17857: Procedure FormKeyPress( Sender : TObject; var Key : Char)
17858: procedure FormOutput1Click(Sender: TObject);
17859: Procedure FormToHtml( Form : TForm; Path : string)
17860: procedure FrameRect(const Rect: TRect);
17861: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
17862: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
17863: Procedure Free( Buffer : TRecordBuffer)
17864: Procedure Free( Buffer : TValueBuffer)
17865: Procedure Free;
17866: Procedure FreeAndNil(var Obj:TObject)
17867: Procedure FreeImage
17868: procedure FreeMem(P: PChar; Size: Integer)
17869: Procedure FreeTreeData( Tree : TUpdateTree)
17870: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
17871: Procedure FullCollapse
17872: Procedure FullExpand
17873: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
17874: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
17875: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
```

```
17876: Procedure Get1( AURL : string; const AResponseContent : TStream);
17877: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
17878: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
17879: Procedure GetAliasNames( List : TStrings)
17880: Procedure GetAliasParams( const AliasName : string; List : TStrings)
17881: Procedure GetApplicationsRunning( Strings : TStrings)
17882: Procedure getBox(aURL, extension: string);
17883: Procedure GetCommandTypes( List : TWideStrings)
17884: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
17885: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
17886: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
17887: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
17888: Procedure GetDatabaseNames( List : TStrings)
17889: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
17890: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
17891: Procedure GetDir(d: byte; var s: string)
17892: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
17893: Procedure GetDriverNames( List : TStrings)
17894: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
17895: Procedure GetDriverParams( const DriverName : string; List : TStrings)
17896: Procedure GetEMails1Click( Sender : TObject)
17897: Procedure getEnvironmentInfo;
17898: Function getEnvironmentString: string;
17899: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
17900: Procedure GetFieldNames( const TableName : string; List : TStrings)
17901: Procedure GetFieldNames( const TableName : string; List : TStrings);
17902: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
17903: Procedure GETFIELDNAMES( LIST : TSTRINGS)
17904: Procedure GetFieldNames1( const TableName : string; List : TStrings);
17905: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
17906: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
17907: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
17908: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
17909: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
17910: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
17911: Procedure GetFormatSettings
17912: Procedure GetFromDIB( var DIB : TBitmapInfo)
17913: Procedure GetFromHDIB( HDIB : HBitmap)
17914: Procedure GetIcon( Index : Integer; Image : TIcon);
17915: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
17916: Procedure GetIndexInfo( IndexName : string)
17917: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
17918: Procedure GetIndexNames( List : TStrings)
17919: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
17920: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
17921: Procedure GetIndexNames4( const TableName : string; List : TStrings);
17922: Procedure GetInternalResponse
17923: Procedure GETITEMNAMES( LIST : TSTRINGS)
17924: procedure GetMem(P: PChar; Size: Integer)
17925: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
17926: procedure GetPackageDescription(ModuleName: PChar): string)
17927: Procedure GetPackageNames( List : TStrings);
17928: Procedure GetPackageNames1( List : TWideStrings);
17929: Procedure GetParamList( List : TList; const ParamNames : WideString)
17930: Procedure GetProcedureNames( List : TStrings);
17931: Procedure GetProcedureNames( List : TWideStrings);
17932: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
17933: Procedure GetProcedureNames1( List : TStrings);
17934: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
17935: Procedure GetProcedureNames3( List : TWideStrings);
17936: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
17937: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
17938: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
17939: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
17940: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
17941: Procedure GetProviderNames( Names : TWideStrings);
17942: Procedure GetProviderNames( Proc : TGetStrProc)
17943: Procedure GetProviderNames1( Names : TStrings);
17944: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
17945: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no open image
17946: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
17947: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
17948: Procedure GetSchemaNames( List : TStrings);
17949: Procedure GetSchemaNames1( List : TWideStrings);
17950: Procedure getScriptandRunAsk;
17951: Procedure getScriptandRun(ascript: string);
17952: Procedure getScript(ascript: string);   //alias
17953: Procedure getWebScript(ascript: string);   //alias
17954: Procedure GetSessionNames( List : TStrings)
17955: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
17956: Procedure GetStrings( List : TStrings)
17957: Procedure GetSystemTime; stdcall;
17958: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
17959: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
17960: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
17961: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
17962: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
17963: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
```

```
17964: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
17965: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
17966: Procedure GetVisibleWindows( List : Tstrings)
17967: Procedure GoBegin
17968: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
17969: Procedure GotoCurrent( Table : TTable)
17970: procedure GotoEnd1Click(Sender: TObject);
17971: Procedure GotoNearest
17972: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
17973: Procedure HandleException( E : Exception; var Handled : Boolean)
17974: procedure HANDLEMESSAGE
17975: procedure HandleNeeded;
17976: Procedure Head( AURL : string)
17977: Procedure Help( var AHelpContents : TStringList; ACommand : String)
17978: Procedure HexToBinary( Stream : TStream)
17979: procedure HexToBinary(Stream:TStream)
17980: Procedure HideDragImage
17981: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
17982: Procedure HideTraybar
17983: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
17984: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
17985: Procedure HookOSExceptions
17986: Procedure HookSignal( RtlSigNum : Integer)
17987: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
17988: Procedure HTMLSyntax1Click( Sender : TObject)
17989: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
17990: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
17991: Procedure ImportfromClipboard1Click( Sender : TObject)
17992: Procedure ImportfromClipboard2Click( Sender : TObject)
17993: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
17994: procedure Incb(var x: byte);
17995: Procedure Include1Click( Sender : TObject)
17996: Procedure IncludeOFF;  //preprocessing
17997: Procedure IncludeON;
17998: procedure Info1Click(Sender: TObject);
17999: Procedure InitAltRecBuffers( CheckModified : Boolean)
18000: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
18001: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
18002: Procedure InitData( ASource : TDataSet)
18003: Procedure InitDelta( ADelta : TPacketDataSet);
18004: Procedure InitDelta1( const ADelta : OleVariant);
18005: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
18006: Procedure Initialize
18007: procedure InitializePackage(Module: HMODULE)
18008: Procedure INITIATEACTION
18009: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
18010: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
18011: Procedure InitModule( AModule : TComponent)
18012: Procedure InitStdConvs
18013: Procedure InitTreeData( Tree : TUpdateTree)
18014: Procedure INSERT
18015: Procedure Insert( Index : Integer; AClass : TClass)
18016: Procedure Insert( Index : Integer; AComponent : TComponent)
18017: Procedure Insert( Index : Integer; AObject : TObject)
18018: Procedure Insert( Index : Integer; const S : WideString)
18019: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
18020: Procedure Insert(Index: Integer; const S: string);
18021: procedure Insert(Index: Integer; S: string);
18022: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
18023: procedure InsertComponent(AComponent:TComponent)
18024: procedure InsertControl(AControl: TControl);
18025: Procedure InsertIcon( Index : Integer; Image : TIcon)
18026: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
18027: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
18028: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
18029: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
18030: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
18031: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
18032: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
18033: Procedure InternalBeforeResolve( Tree : TUpdateTree)
18034: Procedure InvalidateModuleCache
18035: Procedure InvalidateTitles
18036: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
18037: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
18038: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
18039: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
18040: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
18041: procedure JavaSyntax1Click(Sender: TObject);
18042: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
18043: Procedure KillDataChannel
18044: Procedure Largefont1Click( Sender : TObject)
18045: Procedure LAST
18046: Procedure LaunchCpl( FileName : string)
18047: Procedure Launch( const AFile : string)
18048: Procedure LaunchFile( const AFile : string)
18049: Procedure LetFileList(FileList: TStringlist; apath: string);
18050: Procedure lineToNumber( xmemo : String; met : boolean)
```

```
18051: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
18052: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
18053: Procedure ListViewData( Sender : TObject; Item : TListItem)
18054: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
18055: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
18056: Procedure ListViewDblClick( Sender : TObject)
18057: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18058: Procedure ListDLLExports(const FileName: string; List: TStrings);
18059: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
18060: procedure LoadBytecode1Click(Sender: TObject);
18061: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
18062: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
18063: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
18064: Procedure LoadFromFile( AFileName : string)
18065: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
18066: Procedure LoadFromFile( const FileName : string)
18067: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
18068: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
18069: Procedure LoadFromFile( const FileName : WideString)
18070: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18071: Procedure LoadFromFile(const AFileName: string)
18072: procedure LoadFromFile(FileName: string);
18073: procedure LoadFromFile(FileName:String)
18074: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
18075: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
18076: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
18077: Procedure LoadFromStream( const Stream : TStream)
18078: Procedure LoadFromStream( S : TStream)
18079: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18080: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18081: Procedure LoadFromStream( Stream : TStream)
18082: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
18083: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
18084: procedure LoadFromStream(Stream: TStream);
18085: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
18086: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
18087: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
18088: Procedure LoadLastFile1Click( Sender : TObject)
18089: { LoadIcoToImage loads two icons from resource named NameRes,
18090:   into two image lists ALarge and ASmall}
18091: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
18092: Procedure LoadMemo
18093: Procedure LoadParamsFromIniFile( FFileName : WideString)
18094: Procedure Lock
18095: Procedure Login
18096: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
18097: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
18098: Procedure MakeCaseInsensitive
18099: Procedure MakeDeterministic( var bChanged : boolean)
18100: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
18101: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
18102: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
18103: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
18104: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
18105: Procedure SetRectComplexFormatStr( const S : string)
18106: Procedure SetPolarComplexFormatStr( const S : string)
18107: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
18108: Procedure MakeVisible
18109: Procedure MakeVisible( PartialOK : Boolean)
18110: Procedure Manual1Click( Sender : TObject)
18111: Procedure MarkReachable
18112: Procedure maXbox;  //shows the exe version data in a win box
18113: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
18114: Procedure Memo1Change( Sender : TObject)
18115: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
18116: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
18117: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
18118: procedure Memory1Click(Sender: TObject);
18119: Procedure MERGE( MENU : TMAINMENU)
18120: Procedure MergeChangeLog
18121: procedure MINIMIZE
18122: Procedure MinimizeMaxbox;
18123: Procedure MkDir(const s: string)
18124: Procedure MakeDir(const s: string)');
18125: Procedure ChangeDir(const s: string)');
18126: Function makeFile(const FileName: string): integer)');
18127: Procedure mnuPrintFont1Click( Sender : TObject)
18128: procedure ModalStarted
18129: Procedure Modified
18130: Procedure ModifyAlias( Name : string; List : TStrings)
18131: Procedure ModifyDriver( Name : string; List : TStrings)
18132: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
18133: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
18134: Procedure Move( CurIndex, NewIndex : Integer)
18135: procedure Move(CurIndex, NewIndex: Integer);
```

```
18136: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
18137: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
18138: Procedure moveCube( o : TMyLabel)
18139: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
18140: procedure MoveTo(X, Y: Integer);
18141: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
18142: Procedure MovePoint(var x,y:Extended; const angle:Extended);
18143: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
18144: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
18145: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
18146: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
18147: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
18148: procedure New(P: PChar)
18149: procedure New1Click(Sender: TObject);
18150: procedure NewInstance1Click(Sender: TObject);
18151: Procedure NEXT
18152: Procedure NextMonth
18153: Procedure Noop
18154: Procedure NormalizePath( var APath : string)
18155: procedure ObjectBinaryToText(Input, Output: TStream)
18156: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18157: procedure ObjectResourceToText(Input, Output: TStream)
18158: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18159: procedure ObjectTextToBinary(Input, Output: TStream)
18160: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18161: procedure ObjectTextToResource(Input, Output: TStream)
18162: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18163: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
18164: Procedure Open( const UserID : WideString; const Password : WideString);
18165: Procedure Open;
18166: Procedure open1Click( Sender : TObject)
18167: Procedure OpenCdDrive
18168: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
18169: Procedure OpenCurrent
18170: Procedure OpenFile(vfilenamepath: string)
18171: Procedure OpenDirectory1Click( Sender : TObject)
18172: Procedure OpenDir(adir: string);
18173: Procedure OpenIndexFile( const IndexName : string)
18174: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
18175: Procedure OpenWriteBuffer( const AThreshhold : Integer)
18176: Procedure OptimizeMem
18177: Procedure Options1( AURL : string);
18178: Procedure OutputDebugString(lpOutputString : PChar)
18179: Procedure PackBuffer
18180: Procedure Paint
18181: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
18182: Procedure PaintToTBitmap( Target : TBitmap)
18183: Procedure PaletteChanged
18184: Procedure ParentBiDiModeChanged
18185: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
18186: Procedure PasteFromClipboard;
18187: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
18188: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
18189: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
18190: Procedure PError( Text : string)
18191: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18192: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
18193: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
18194: procedure playmp3(mpath: string);
18195: Procedure PlayMP31Click( Sender : TObject)
18196: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
18197: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
18198: procedure PolyBezier(const Points: array of TPoint);
18199: procedure PolyBezierTo(const Points: array of TPoint);
18200: procedure Polygon(const Points: array of TPoint);
18201: procedure Polyline(const Points: array of TPoint);
18202: Procedure Pop
18203: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
18204: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
18205: Procedure POPUP( X, Y : INTEGER)
18206: Procedure PopupURL(URL : WideString);
18207: Procedure POST
18208: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
18209: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
18210: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
18211: Procedure PostUser( const Email, FirstName, LastName : WideString)
18212: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
18213: procedure Pred(X: int64);
18214: Procedure Prepare
18215: Procedure PrepareStatement
18216: Procedure PreProcessXML( AList : TStrings)
18217: Procedure PreventDestruction
18218: Procedure Print( const Caption : string)
18219: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
18220: procedure printf(const format: String; const args: array of const);
18221: Procedure PrintList(Value: TStringList);
18222: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
18223: Procedure Printout1Click( Sender : TObject)
```

```
18224: Procedure ProcessHeaders
18225: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
18226: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
18227: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
18228: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
18229: Procedure ProcessMessagesOFF;  //application.processmessages
18230: Procedure ProcessMessagesON;
18231: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
18232: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
18233: Procedure Proclist Size is: 3797 /1415
18234: Procedure procMessClick( Sender : TObject)
18235: Procedure PSScriptCompile( Sender : TPSScript)
18236: Procedure PSScriptExecute( Sender : TPSScript)
18237: Procedure PSScriptLine( Sender : TObject)
18238: Procedure Push( ABoundary : string)
18239: procedure PushItem(AItem: Pointer)
18240: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
18241: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
18242: procedure PutLinuxLines(const Value: string)
18243: Procedure Quit
18244: Procedure RaiseConversionError( const AText : string);
18245: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
18246: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
18247: procedure RaiseException(Ex: TIFException; Param: String);
18248: Procedure RaiseExceptionForLastCmdResult;
18249: procedure RaiseLastException;
18250: procedure RaiseException2;
18251: Procedure RaiseException3(const Msg: string);
18252: Procedure RaiseExcept(const Msg: string);
18253: Procedure RaiseLastOSError
18254: Procedure RaiseLastWin32;
18255: procedure RaiseLastWin32Error)
18256: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
18257: Procedure RandomFillStream( Stream : TMemoryStream)
18258: procedure randomize;
18259: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
18260: Procedure RCS
18261: Procedure Read( Socket : TSocket)
18262: Procedure ReadBlobData
18263: procedure ReadBuffer(Buffer:String;Count:LongInt)
18264: procedure ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
18265: Procedure ReadSection( const Section : string; Strings : TStrings)
18266: Procedure ReadSections( Strings : TStrings)
18267: Procedure ReadSections( Strings : TStrings);
18268: Procedure ReadSections1( const Section : string; Strings : TStrings);
18269: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
18270: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
18271: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
18272: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
18273: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
18274: Procedure Realign;
18275: procedure Rectangle(X1, Y1, X2, Y2: Integer);
18276: Procedure Rectangle1( const Rect : TRect);
18277: Procedure RectCopy( var Dest : TRect; const Source : TRect)
18278: Procedure RectFitToScreen( var R : TRect)
18279: Procedure RectGrow( var R : TRect; const Delta : Integer)
18280: Procedure RectGrowX( var R : TRect; const Delta : Integer)
18281: Procedure RectGrowY( var R : TRect; const Delta : Integer)
18282: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
18283: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
18284: Procedure RectNormalize( var R : TRect)
18285: //  TFileCallbackProcedure = procedure(filename:string);
18286: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
18287: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
18288: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
18289: Procedure Refresh;
18290: Procedure RefreshData( Options : TFetchOptions)
18291: Procedure REFRESHLOOKUPLIST
18292: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
18293: Procedure RegExPathfinder2(Pathin, fileout, firstp, aregex, ext: string; asort, acopy: boolean);
18294: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
18295: Procedure RegisterChanges( Value : TChangeLink)
18296: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
18297: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
18298: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
18299: Procedure ReInitialize( ADelay : Cardinal)
18300: procedure RELEASE
18301: Procedure Remove( const AByteCount : integer)
18302: Procedure REMOVE( FIELD : TFIELD)
18303: Procedure REMOVE( ITEM : TMENUITEM)
18304: Procedure REMOVE( POPUP : TPOPUPMENU)
18305: Procedure RemoveAllPasswords
18306: procedure RemoveComponent(AComponent:TComponent)
18307: Procedure RemoveDir( const ADirName : string)
18308: Procedure RemoveLambdaTransitions( var bChanged : boolean)
18309: Procedure REMOVEPARAM( VALUE : TPARAM)
18310: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
18311: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
18312: Procedure Rename( const ASourceFile, ADestFile : string)
```

```
18313: Procedure Rename( const FileName : string; Reload : Boolean)
18314: Procedure RenameTable( const NewTableName : string)
18315: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
18316: Procedure Replace1Click( Sender : TObject)
18317: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
18318: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
18319: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
18320: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
18321: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
18322: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
18323: Procedure Requery( Options : TExecuteOptions)
18324: Procedure Reset
18325: Procedure Reset1Click( Sender : TObject)
18326: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
18327: procedure ResourceExplore1Click(Sender: TObject);
18328: Procedure RestoreContents
18329: Procedure RestoreDefaults
18330: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
18331: Procedure RetrieveHeaders
18332: Procedure RevertRecord
18333: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
18334: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18335: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18336: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
18337: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
18338: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
18339: Procedure RleCompress2( Stream : TStream)
18340: Procedure RleDecompress2( Stream : TStream)
18341: Procedure RmDir(const S: string)
18342: Procedure Rollback
18343: Procedure Rollback( TransDesc : TTransactionDesc)
18344: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
18345: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
18346: Procedure RollbackTrans
18347: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
18348: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
18349: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
18350: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
18351: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
18352: Procedure S_EBox( const AText : string)
18353: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
18354: Procedure S_IBox( const AText : string)
18355: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
18356: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
18357: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
18358: Procedure SampleVarianceAndMean
18359: ( const X : TDynFloatArray; var Variance, Mean : Float)
18360: Procedure Save2Click( Sender : TObject)
18361: Procedure Saveas3Click( Sender : TObject)
18362: Procedure Savebefore1Click( Sender : TObject)
18363: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
18364: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
18365: Procedure SaveConfigFile
18366: Procedure SaveOutput1Click( Sender : TObject)
18367: procedure SaveScreenshotClick(Sender: TObject);
18368: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
18369: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
18370: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
18371: Procedure SaveToFile( AFileName : string)
18372: Procedure SAVETOFILE( const FILENAME : String)
18373: Procedure SaveToFile( const FileName : WideString)
18374: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
18375: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18376: procedure SaveToFile(FileName: string);
18377: procedure SaveToFile(FileName:String)
18378: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
18379: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18380: Procedure SaveToStream( S : TStream)
18381: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18382: Procedure SaveToStream( Stream : TStream)
18383: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
18384: procedure SaveToStream(Stream: TStream);
18385: procedure SaveToStream(Stream:TStream)
18386: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
18387: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
18388: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
18389: procedure Say(const sText: string)
18390: Procedure SBytecode1Click( Sender : TObject)
18391: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
18392: procedure ScriptExplorer1Click(Sender: TObject);
18393: Procedure Scroll( Distance : Integer)
18394: Procedure Scroll( DX, DY : Integer)
18395: procedure ScrollBy(DeltaX, DeltaY: Integer);
18396: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
18397: Procedure ScrollTabs( Delta : Integer)
18398: Procedure Search1Click( Sender : TObject)
18399: procedure SearchAndOpenDoc(vfilenamepath: string)
18400: procedure SearchAndOpenFile(vfilenamepath: string)
```

```
18401: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
18402: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
18403: Procedure SearchNext1Click( Sender : TObject)
18404: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
18405: Procedure Select1( const Nodes : array of TTreeNode);
18406: Procedure Select2( Nodes : TList);
18407: Procedure SelectNext( Direction : Boolean)
18408: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
18409: Procedure SelfTestPEM  //unit uPSI_cPEM
18410: Procedure Send( AMsg : TIdMessage)
18411: //config forst in const MAILINIFILE = 'maildef.ini';
18412: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
18413: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18414: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18415: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
18416: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
18417: Procedure SendResponse
18418: Procedure SendStream( AStream : TStream)
18419: Procedure Set8087CW( NewCW : Word)
18420: Procedure SetAll( One, Two, Three, Four : Byte)
18421: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
18422: Procedure SetAppDispatcher( const ADispatcher : TComponent)
18423: procedure SetArrayLength;
18424: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
18425: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
18426: Procedure SetAsHandle( Format : Word; Value : THandle)
18427: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
18428: procedure SetCaptureControl(Control: TControl);
18429: Procedure SetColumnAttributes
18430: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
18431: Procedure SetCustomHeader( const Name, Value : string)
18432: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
18433: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
18434: Procedure SetFocus
18435: procedure SetFocus; virtual;
18436: Procedure SetInitialState
18437: Procedure SetKey
18438: procedure SetLastError(ErrorCode: Integer)
18439: procedure SetLength;
18440: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
18441: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
18442: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
18443: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
18444: Procedure SetParams1( UpdateKind : TUpdateKind);
18445: Procedure SetPassword( const Password : string)
18446: Procedure SetPointer( Ptr : Pointer; Size : Longint)
18447: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
18448: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
18449: Procedure SetProvider( Provider : TComponent)
18450: Procedure SetProxy( const Proxy : string)
18451: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
18452: Procedure SetRange( const StartValues, EndValues : array of const)
18453: Procedure SetRangeEnd
18454: Procedure SetRate( const aPercent, aYear : integer)
18455: procedure SetRate(const aPercent, aYear: integer)
18456: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
18457: Procedure SetSafeCallExceptionMsg( Msg : String)
18458: procedure SETSELTEXTBUF(BUFFER:PCHAR)
18459: Procedure SetSize( AWidth, AHeight : Integer)
18460: procedure SetSize(NewSize:LongInt)
18461: procedure SetString(var s: string; buffer: PChar; len: Integer)
18462: Procedure SetStrings( List : TStrings)
18463: Procedure SetText( Text : PwideChar)
18464: procedure SetText(Text: PChar);
18465: Procedure SetTextBuf( Buffer : PChar)
18466: procedure SETTEXTBUF(BUFFER:PCHAR)
18467: Procedure SetTick( Value : Integer)
18468: Procedure SetTimeout( ATimeOut : Integer)
18469: Procedure SetTraceEvent( Event : TDBXTraceEvent)
18470: Procedure SetUserName( const UserName : string)
18471: Procedure SetWallpaper( Path : string);
18472: procedure ShellStyle1Click(Sender: TObject);
18473: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
18474: Procedure ShowFileProperties( const FileName : string)
18475: Procedure ShowInclude1Click( Sender : TObject)
18476: Procedure ShowInterfaces1Click( Sender : TObject)
18477: Procedure ShowLastException1Click( Sender : TObject)
18478: Procedure ShowMessage( const Msg : string)
18479: Procedure ShowMessageBig(const aText : string);
18480: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
18481: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
18482: Procedure MsgBig(const aText : string);        //alias
18483: procedure showmessage(mytext: string);
18484: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
18485: procedure ShowMessageFmt(const Msg: string; Params: array of const))
18486: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
18487: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
18488: Procedure ShowSearchDialog( const Directory : string)
```

```
18489: Procedure ShowSpecChars1Click( Sender : TObject)
18490: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
18491: Procedure ShredFile( const FileName : string; Times : Integer)
18492: procedure Shuffle(vQ: TStringList);
18493: Procedure ShuffleList( var List : array of Integer; Count : Integer)
18494: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
18495: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
18496: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
18497: Procedure Site( const ACommand : string)
18498: Procedure SkipEOL
18499: Procedure Sleep( ATime : cardinal)
18500: Procedure Sleep( milliseconds : Cardinal)
18501: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
18502: Procedure Slinenumbers1Click( Sender : TObject)
18503: Procedure Sort
18504: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
18505: procedure Speak(const sText: string)   //async like voice
18506: procedure Speak2(const sText: string)  //sync
18507: procedure Split(Str: string;  SubStr: string; List: TStrings);
18508: Procedure SplitNameValue( const Line : string; var Name, Value : string)
18509: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
18510: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
18511: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
18512: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
18513: procedure SQLSyntax1Click(Sender: TObject);
18514: Procedure SRand( Seed : RNG_IntType)
18515: Procedure Start
18516: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
18517: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
18518: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
18519: Procedure StartTransaction( TransDesc : TTransactionDesc)
18520: Procedure Status( var AStatusList : TStringList)
18521: Procedure StatusBar1DblClick( Sender : TObject)
18522: Procedure StepInto1Click( Sender : TObject)
18523: Procedure StepIt
18524: Procedure StepOut1Click( Sender : TObject)
18525: Procedure Stop
18526: procedure stopmp3;
18527: procedure StartWeb(aurl: string);
18528: Procedure Str(aint: integer; astr: string); //of system
18529: Procedure StrDispose( Str : PChar)
18530: procedure StrDispose(Str: PChar)
18531: Procedure StrReplace(var Str: String; Old, New: String);
18532: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
18533: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
18534: Procedure StringToBytes( Value : String; Bytes : array of byte)
18535: procedure StrSet(c : Char; I : Integer; var s : String);
18536: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
18537: Procedure StructureMount( APath : String)
18538: procedure STYLECHANGED(SENDER:TOBJECT)
18539: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
18540: procedure Succ(X: int64);
18541: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
18542: procedure SwapChar(var X,Y: char);  //swapX follows
18543: Procedure SwapFloats( var X, Y : Float)
18544: procedure SwapGrid(grd: TStringGrid);
18545: Procedure SwapOrd( var I, J : Byte);
18546: Procedure SwapOrd( var X, Y : Integer)
18547: Procedure SwapOrd1( var I, J : Shortint);
18548: Procedure SwapOrd2( var I, J : Smallint);
18549: Procedure SwapOrd3( var I, J : Word);
18550: Procedure SwapOrd4( var I, J : Integer);
18551: Procedure SwapOrd5( var I, J : Cardinal);
18552: Procedure SwapOrd6( var I, J : Int64);
18553: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
18554: Procedure Synchronize1( Method : TMethod);
18555: procedure SyntaxCheck1Click(Sender: TObject);
18556: Procedure SysFreeString(const S: WideString); stdcall;
18557: Procedure TakeOver( Other : TLinearBitmap)
18558: Procedure Talkln(const sText: string)  //async voice
18559: procedure tbtn6resClick(Sender: TObject);
18560: Procedure tbtnUseCaseClick( Sender : TObject)
18561: procedure TerminalStyle1Click(Sender: TObject);
18562: Procedure Terminate
18563: Procedure texSyntax1Click( Sender : TObject)
18564: procedure TextOut(X, Y: Integer; Text: string);
18565: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
18566: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
18567: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
18568: Procedure TextStart
18569: procedure TILE
18570: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
18571: Procedure TitleClick( Column : TColumn)
18572: Procedure ToDo
18573: procedure toolbtnTutorialClick(Sender: TObject);
18574: Procedure Trace1( AURL : string; const AResponseContent : TStream);
18575: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
18576: Procedure Truncate
18577: procedure Tutorial101Click(Sender: TObject);
```

```
18578: procedure Tutorial10Statistics1Click(Sender: TObject);
18579: procedure Tutorial11Forms1Click(Sender: TObject);
18580: procedure Tutorial12SQL1Click(Sender: TObject);
18581: Procedure tutorial1Click( Sender : TObject)
18582: Procedure tutorial21Click( Sender : TObject)
18583: Procedure tutorial31Click( Sender : TObject)
18584: Procedure tutorial4Click( Sender : TObject)
18585: Procedure Tutorial5Click( Sender : TObject)
18586: procedure Tutorial6Click(Sender: TObject);
18587: procedure Tutorial91Click(Sender: TObject);
18588: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
18589: procedure UniqueString(var str: AnsiString)
18590: procedure UnloadLoadPackage(Module: HMODULE)
18591: Procedure Unlock
18592: Procedure UNMERGE( MENU : TMAINMENU)
18593: Procedure UnRegisterChanges( Value : TChangeLink)
18594: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
18595: Procedure UnregisterConversionType( const AType : TConvType)
18596: Procedure UnRegisterProvider( Prov : TCustomProvider)
18597: Procedure UPDATE
18598: Procedure UpdateBatch( AffectRecords : TAffectRecords)
18599: Procedure UPDATECURSORPOS
18600: Procedure UpdateFile
18601: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
18602: Procedure UpdateResponse( AResponse : TWebResponse)
18603: Procedure UpdateScrollBar
18604: Procedure UpdateView1Click( Sender : TObject)
18605: procedure Val(const s: string; var n, z: Integer)
18606: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
18607: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
18608: Procedure VariantAdd( const src : Variant; var dst : Variant)
18609: Procedure VariantAnd( const src : Variant; var dst : Variant)
18610: Procedure VariantArrayRedim( var V : Variant; High : Integer)
18611: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
18612: Procedure VariantClear( var V : Variant)
18613: Procedure VariantCpy( const src : Variant; var dst : Variant)
18614: Procedure VariantDiv( const src : Variant; var dst : Variant)
18615: Procedure VariantMod( const src : Variant; var dst : Variant)
18616: Procedure VariantMul( const src : Variant; var dst : Variant)
18617: Procedure VariantOr( const src : Variant; var dst : Variant)
18618: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
18619: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
18620: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
18621: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
18622: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
18623: Procedure VariantShl( const src : Variant; var dst : Variant)
18624: Procedure VariantShr( const src : Variant; var dst : Variant)
18625: Procedure VariantSub( const src : Variant; var dst : Variant)
18626: Procedure VariantXor( const src : Variant; var dst : Variant)
18627: Procedure VarCastError;
18628: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
18629: Procedure VarInvalidOp
18630: Procedure VarInvalidNullOp
18631: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
18632: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
18633: Procedure VarArrayCreateError
18634: Procedure VarResultCheck( AResult : HRESULT);
18635: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
18636: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
18637: Function VarTypeAsText( const AType : TVarType) : string
18638: procedure Voice(const sText: string) //async
18639: procedure Voice2(const sText: string) //sync
18640: Procedure WaitMiliSeconds( AMSec : word)
18641: Procedure WaitMS( AMSec : word)');
18642: procedure WebCamPic(picname: string);  //eg: c:\mypic.png
18643: Procedure WideAppend( var dst : WideString; const src : WideString)
18644: Procedure WideAssign( var dst : WideString; var src : WideString)
18645: Procedure WideDelete( var dst : WideString; index, count : Integer)
18646: Procedure WideFree( var s : WideString)
18647: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
18648: Procedure WideFromPChar( var dst : WideString; src : PChar)
18649: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
18650: Procedure WideSetLength( var dst : WideString; len : Integer)
18651: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
18652: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
18653: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
18654: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
18655: Procedure HttpGet(const Url: string; Stream:TStream);
18656: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
18657: Procedure WordWrap1Click( Sender : TObject)
18658: Procedure Write( const AOut : string)
18659: Procedure Write( Socket : TSocket)
18660: procedure Write(S: string);
18661: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
18662: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
18663: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
18664: procedure WriteBuffer(Buffer:String;Count:LongInt)
18665: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
18666: Procedure WriteChar( AValue : Char)
```

```
18667: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
18668: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
18669: Procedure WriteFloat( const Section, Name : string; Value : Double)
18670: Procedure WriteHeader( AHeader : TStrings)
18671: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
18672: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
18673: Procedure WriteLn( const AOut : string)
18674: procedure Writeln(s: string);
18675: Procedure WriteLog( const FileName, LogLine : string)
18676: Procedure WriteRFCReply( AReply : TIdRFCReply)
18677: Procedure WriteRFCStrings( AStrings : TStrings)
18678: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
18679: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
18680: Procedure WriteString( const Section, Ident, Value : String)
18681: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
18682: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
18683: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
18684: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18685: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18686: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
18687: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
18688: procedure XMLSyntax1Click(Sender: TObject);
18689: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
18690: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
18691: Procedure ZeroFillStream( Stream : TMemoryStream)
18692: procedure XMLSyntax1Click(Sender: TObject);
18693: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
18694: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
18695: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
18696: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
18697: procedure(Sender, Source: TObject;X, Y: Integer)
18698: procedure(Sender, Target: TObject; X, Y: Integer)
18699: procedure(Sender: TObject; ASection, AWidth: Integer)
18700: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
18701: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
18702: procedure(Sender: TObject; var Action: TCloseAction)
18703: procedure(Sender: TObject; var CanClose: Boolean)
18704: procedure(Sender: TObject; var Key: Char);
18705: ProcedureName ProcedureNames ProcedureParametersCursor @
18706:
18707: *************Now Constructors constructor *************
18708:  Size is: 1248 1115 996 628 550 544 501 459 (381)
18709:  Attach( VersionInfoData : Pointer; Size : Integer)
18710:  constructor Create( ABuckets : TBucketListSizes)
18711:  Create( ACallBackWnd : HWND)
18712:  Create( AClient : TCustomTaskDialog)
18713:  Create( AClient : TIdTelnet)
18714:  Create( ACollection : TCollection)
18715:  Create( ACollection : TFavoriteLinkItems)
18716:  Create( ACollection : TTaskDialogButtons)
18717:  Create( AConnection : TIdCustomHTTP)
18718:  Create( ACreateSuspended : Boolean)
18719:  Create( ADataSet : TCustomSQLDataSet)
18720:  CREATE( ADATASET : TDATASET)
18721:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
18722:  Create( AGrid : TCustomDBGrid)
18723:  Create( AGrid : TStringGrid; AIndex : Longint)
18724:  Create( AHTTP : TIdCustomHTTP)
18725:  Create( AListItems : TListItems)
18726:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
18727:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
18728:  Create( AOwner : TCommonCalendar)
18729:  Create( AOwner : TComponent)
18730:  CREATE( AOWNER : TCOMPONENT)
18731:  Create( AOwner : TCustomListView)
18732:  Create( AOwner : TCustomOutline)
18733:  Create( AOwner : TCustomRichEdit)
18734:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
18735:  Create( AOwner : TCustomTreeView)
18736:  Create( AOwner : TIdUserManager)
18737:  Create( AOwner : TListItems)
18738:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
18739:  CREATE( AOWNER : TPERSISTENT)
18740:  Create( AOwner : TPersistent)
18741:  Create( AOwner : TTable)
18742:  Create( AOwner : TTreeNodes)
18743:  Create( AOwner : TWinControl; const ClassName : string)
18744:  Create( AParent : TIdCustomHTTP)
18745:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
18746:  Create( AProvider : TBaseProvider)
18747:  Create( AProvider : TCustomProvider);
18748:  Create( AProvider : TDataSetProvider)
18749:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
18750:  Create( ASocket : TSocket)
18751:  Create( AStrings : TWideStrings)
18752:  Create( AToolBar : TToolBar)
18753:  Create( ATreeNodes : TTreeNodes)
18754:  Create( Autofill : boolean)
18755:  Create( AWebPageInfo : TAbstractWebPageInfo)
```

```
18756:  Create( AWebRequest : TWebRequest)
18757:  Create( Collection : TCollection)
18758:  Create( Collection : TIdMessageParts; ABody : TStrings)
18759:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
18760:  Create( Column : TColumn)
18761:  Create( const AConvFamily : TConvFamily; const ADescription : string)
18762:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
18763:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
18764:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
18765:  Create( const ATabSet : TTabSet)
18766:  Create( const Compensate : Boolean)
18767:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
18768:  Create( const FileName : string)
18769:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
18770:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
18771:  Create( const MaskValue : string)
18772:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
18773:  Create( const Prefix : string)
18774:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
18775:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18776:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18777:  Create( CoolBar : TCoolBar)
18778:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
18779:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
18780:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
18781:  Create( DBCtrlGrid : TDBCtrlGrid)
18782:  Create( DSTableProducer : TDSTableProducer)
18783:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
18784:  Create( ErrorCode : DBIResult)
18785:  Create( Field : TBlobField; Mode : TBlobStreamMode)
18786:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
18787:  Create( HeaderControl : TCustomHeaderControl)
18788:  Create( HTTPRequest : TWebRequest)
18789:  Create( iStart : integer; sText : string)
18790:  Create( iValue : Integer)
18791:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
18792:  Create( MciErrNo : MCIERROR; const Msg : string)
18793:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
18794:  Create( Message : string; ErrorCode : DBResult)
18795:  Create( Msg : string)
18796:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
18797:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
18798:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
18799:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
18800:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
18801:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
18802:  Create( Owner : TCustomComboBoxEx)
18803:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
18804:  Create( Owner : TPersistent)
18805:  Create( Params : TStrings)
18806:  Create( Size : Cardinal)
18807:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
18808:  Create( StatusBar : TCustomStatusBar)
18809:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
18810:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
18811:  Create(AHandle:Integer)
18812:  Create(AOwner: TComponent); virtual;
18813:  Create(const AURI : string)
18814:  Create(FileName:String;Mode:Word)
18815:  Create(Instance:THandle;ResName:String;ResType:PChar)
18816:  Create(Stream : TStream)
18817:  Create1( ADataset : TDataset);
18818:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
18819:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
18820:  Create2( Other : TObject);
18821:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
18822:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
18823:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
18824:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
18825:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
18826:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
18827:  CreateRes( Ident : Integer);
18828:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
18829:  CreateRes( ResStringRec : PResStringRec);
18830:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
18831:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
18832:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
18833:  CreateSize( AWidth, AHeight : Integer)
18834:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
18835:
18836: ------------------------------------------------------------------------------
18837: unit uPSI_MathMax;
18838: ------------------------------------------------------------------------------
18839:  CONSTS
```

```
18840:   Bernstein: Float = 0.28016949902386913303643649123 07;  // Bernstein constant
18841:   Cbrt2: Float     = 1.25992104989487316476721060727 82;  // CubeRoot(2)
18842:   Cbrt3: Float     = 1.44224957030740838232163831078 01;  // CubeRoot(3)
18843:   Cbrt10: Float    = 2.15443469003188372175929356651 94;  // CubeRoot(10)
18844:   Cbrt100: Float   = 4.64158883361277889241007635091 94;  // CubeRoot(100)
18845:   CbrtPi: Float    = 1.46459188756152326302014252726 38;  // CubeRoot(PI)
18846:   Catalan: Float   = 0.91596559417721901505460351493 24;  // Catalan constant
18847:   PiJ:   Float       = 3.14159265358979323846264338327 95;  // PI
18848:   PI:  Extended =  3.14159265358979323846264338327 95);
18849:   PiOn2: Float     = 1.57079632679489661923132169163 98;  // PI / 2
18850:   PiOn3: Float     = 1.04719755119659774615421446109 32;  // PI / 3
18851:   PiOn4: Float     = 0.78539816339744830961566084581 988; // PI / 4
18852:   Sqrt2: Float     = 1.41421356237309504880168872420 97;  // Sqrt(2)
18853:   Sqrt3: Float     = 1.73205080756887729352744634150 59;  // Sqrt(3)
18854:   Sqrt5: Float     = 2.23606797749978969640917366873 13;  // Sqrt(5)
18855:   Sqrt10: Float    = 3.16227766016837933199889354443 27;  // Sqrt(10)
18856:   SqrtPi: Float    = 1.77245385090551602729816748334 11;  // Sqrt(PI)
18857:   Sqrt2Pi: Float   = 2.50662827463100050241576528481 1;   // Sqrt(2 * PI)
18858:   TwoPi: Float     = 6.28318530717958647692528676655 9;   // 2 * PI
18859:   ThreePi: Float   = 9.42477796076937971538793014983 85;  // 3 * PI
18860:   Ln2: Float       = 0.69314718055994530941723212145 818; // Ln(2)
18861:   Ln10: Float      = 2.30258509299404568401799145468 44;  // Ln(10)
18862:   LnPi: Float      = 1.14472988584940017414342735135 31;  // Ln(PI)
18863:   Log2J: Float     = 0.30102999566398119521373889472 449; // Log10(2)
18864:   Log3: Float      = 0.47712125471966243729502790325 512; // Log10(3)
18865:   LogPi: Float     = 0.49714987269413385435128828829 09;  // Log10(PI)
18866:   LogE: Float      = 0.43429448190325182765112891891 661; // Log10(E)
18867:   E: Float         = 2.71828182845904523536028747135 27;  // Natural constant
18868:   hLn2Pi: Float    = 0.91893853320467274178032973640 562; // Ln(2*PI)/2
18869:   inv2Pi: Float    = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
18870:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
18871:   GoldenMean: Float   = 1.61803398874989484820458683436563 8;  // GoldenMean
18872:   EulerMascheroni: Float = 0.57721566490153286060651209008 24;  // Euler GAMMA
18873:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
18874:   StDelta         : Extended = 0.00001;  {delta for difference equations}
18875:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
18876:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
18877:
18878: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
18879: begin
18880:   MetersPerInch = 0.0254; // [1]
18881:   MetersPerFoot = MetersPerInch * 12;
18882:   MetersPerYard = MetersPerFoot * 3;
18883:   MetersPerMile = MetersPerFoot * 5280;
18884:   MetersPerNauticalMiles = 1852;
18885:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
18886:   MetersPerLightSecond = 2.99792458E8; // [5]
18887:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
18888:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
18889:   MetersPerCubit = 0.4572; // [6][7]
18890:   MetersPerFathom = MetersPerFoot * 6;
18891:   MetersPerFurlong = MetersPerYard * 220;
18892:   MetersPerHand = MetersPerInch * 4;
18893:   MetersPerPace = MetersPerInch * 30;
18894:   MetersPerRod = MetersPerFoot * 16.5;
18895:   MetersPerChain = MetersPerRod * 4;
18896:   MetersPerLink = MetersPerChain / 100;
18897:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
18898:   MetersPerPica = MetersPerPoint * 12;
18899:
18900:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
18901:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
18902:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
18903:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
18904:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
18905:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
18906:
18907:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
18908:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
18909:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
18910:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
18911:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
18912:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
18913:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
18914:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
18915:
18916:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
18917:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
18918:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
18919:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
18920:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
18921:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
18922:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
18923:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
18924:
18925:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
18926:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
18927:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
18928:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
```

```
18929:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
18930:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
18931:
18932:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
18933:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
18934:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
18935:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
18936:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
18937:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
18938:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
18939:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
18940:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
18941:
18942:   GramsPerPound = 453.59237; // [1][7]
18943:   GramsPerDrams = GramsPerPound / 256;
18944:   GramsPerGrains = GramsPerPound / 7000;
18945:   GramsPerTons = GramsPerPound * 2000;
18946:   GramsPerLongTons = GramsPerPound * 2240;
18947:   GramsPerOunces = GramsPerPound / 16;
18948:   GramsPerStones = GramsPerPound * 14;
18949:
18950:   MaxAngle 9223372036854775808.0;
18951:   MaxTanH 5678.26170314707197474594655389854);
18952:   MaxFactorial( 1754);
18953:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
18954:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
18955:   MaxTanH( 354.89135644669199842162284618659);
18956:   MaxFactorial'LongInt'( 170);
18957:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
18958:   MinFloatingPointD(2.225073858507201383090232717332 4E-308);
18959:   MaxTanH( 44.36141955583649980270285577332 3);
18960:   MaxFactorial'LongInt'( 33);
18961:   MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
18962:   MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
18963:   PiExt( 3.14159265358979323846264338327 95);
18964:   RatioDegToRad( PiExt / 180.0);
18965:   RatioGradToRad( PiExt / 200.0);
18966:   RatioDegToGrad( 200.0 / 180.0);
18967:   RatioGradToDeg( 180.0 / 200.0);
18968:   Crc16PolynomCCITT'LongWord $1021);
18969:  Crc16PolynomIBM'LongWord $8005);
18970:  Crc16Bits'LongInt'( 16);
18971:  Crc16Bytes'LongInt'( 2);
18972:  Crc16HighBit'LongWord $8000);
18973:  NotCrc16HighBit','LongWord $7FFF);
18974:   Crc32PolynomIEEE','LongWord $04C11DB7);
18975:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
18976:  Crc32Koopman','LongWord $741B8CD7);
18977:  Crc32Bits','LongInt'( 32);
18978:  Crc32Bytes','LongInt'( 4);
18979:  Crc32HighBit','LongWord $80000000);
18980:  NotCrc32HighBit','LongWord $7FFFFFFF);
18981:
18982:   MinByte       = Low(Byte);
18983:   MaxByte       = High(Byte);
18984:   MinWord       = Low(Word);
18985:   MaxWord       = High(Word);
18986:   MinShortInt   = Low(ShortInt);
18987:   MaxShortInt   = High(ShortInt);
18988:   MinSmallInt   = Low(SmallInt);
18989:   MaxSmallInt   = High(SmallInt);
18990:   MinLongWord   = LongWord(Low(LongWord));
18991:   MaxLongWord   = LongWord(High(LongWord));
18992:   MinLongInt    = LongInt(Low(LongInt));
18993:   MaxLongInt    = LongInt(High(LongInt));
18994:   MinInt64      = Int64(Low(Int64));
18995:   MaxInt64      = Int64(High(Int64));
18996:   MinInteger    = Integer(Low(Integer));
18997:   MaxInteger    = Integer(High(Integer));
18998:   MinCardinal   = Cardinal(Low(Cardinal));
18999:   MaxCardinal   = Cardinal(High(Cardinal));
19000:   MinNativeUInt = NativeUInt(Low(NativeUInt));
19001:   MaxNativeUInt = NativeUInt(High(NativeUInt));
19002:   MinNativeInt  = NativeInt(Low(NativeInt));
19003:   MaxNativeInt  = NativeInt(High(NativeInt));
19004:  Function CosH( const Z : Float) : Float;
19005:  Function SinH( const Z : Float) : Float;
19006:  Function TanH( const Z : Float) : Float;
19007:
19008:
19009:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
19010:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
19011:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
19012:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
19013:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
19014:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
19015:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
19016:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
19017:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
```

```
19018:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
19019:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
19020:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
19021:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
19022:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
19023:   MachEp = 2.22044604925031308E-16;   { 2^(-52) }
19024:   MaxNum = 1.79769313486231570E+308;  { 2^1024 }
19025:   MinNum = 2.22507385850720138E-308;  { 2^(-1022) }
19026:   MaxLog = 709.782712893384;
19027:   MinLog = -708.396418532264;
19028:   MaxFac = 170;
19029:   MaxGam = 171.624376956302;
19030:   MaxLgm = 2.55634830030E+305;
19031:   SingleCompareDelta   = 1.0E-34;
19032:   DoubleCompareDelta   = 1.0E-280;
19033:   {$IFDEF CLR}
19034:   ExtendedCompareDelta = DoubleCompareDelta;
19035:   {$ELSE}
19036:   ExtendedCompareDelta = 1.0E-4400;
19037:   {$ENDIF}
19038:   Bytes1KB  = 1024;
19039:   Bytes1MB  = 1024 * Bytes1KB;
19040:   Bytes1GB  = 1024 * Bytes1MB;
19041:   Bytes64KB = 64 * Bytes1KB;
19042:   Bytes64MB = 64 * Bytes1MB;
19043:   Bytes2GB  = 2 * LongWord(Bytes1GB);
19044:     clBlack32', $FF000000 ));
19045:     clDimGray32', $FF3F3F3F ));
19046:     clGray32', $FF7F7F7F ));
19047:     clLightGray32', $FFBFBFBF ));
19048:     clWhite32', $FFFFFFFF ));
19049:     clMaroon32', $FF7F0000 ));
19050:     clGreen32', $FF007F00 ));
19051:     clOlive32', $FF7F7F00 ));
19052:     clNavy32', $FF00007F ));
19053:     clPurple32', $FF7F007F ));
19054:     clTeal32', $FF007F7F ));
19055:     clRed32', $FFFF0000 ));
19056:     clLime32', $FF00FF00 ));
19057:     clYellow32', $FFFFFF00 ));
19058:     clBlue32', $FF0000FF ));
19059:     clFuchsia32', $FFFF00FF ));
19060:     clAqua32', $FF00FFFF ));
19061:     clAliceBlue32', $FFF0F8FF ));
19062:     clAntiqueWhite32', $FFFAEBD7 ));
19063:     clAquamarine32', $FF7FFFD4 ));
19064:     clAzure32', $FFF0FFFF ));
19065:     clBeige32', $FFF5F5DC ));
19066:     clBisque32', $FFFFE4C4 ));
19067:     clBlancheDalmond32', $FFFFEBCD ));
19068:     clBlueViolet32', $FF8A2BE2 ));
19069:     clBrown32', $FFA52A2A ));
19070:     clBurlyWood32', $FFDEB887 ));
19071:     clCadetblue32', $FF5F9EA0 ));
19072:     clChartReuse32', $FF7FFF00 ));
19073:     clChocolate32', $FFD2691E ));
19074:     clCoral32', $FFFF7F50 ));
19075:     clCornFlowerBlue32', $FF6495ED ));
19076:     clCornSilk32', $FFFFF8DC ));
19077:     clCrimson32', $FFDC143C ));
19078:     clDarkBlue32', $FF00008B ));
19079:     clDarkCyan32', $FF008B8B ));
19080:     clDarkGoldenRod32', $FFB8860B ));
19081:     clDarkGray32', $FFA9A9A9 ));
19082:     clDarkGreen32', $FF006400 ));
19083:     clDarkGrey32', $FFA9A9A9 ));
19084:     clDarkKhaki32', $FFBDB76B ));
19085:     clDarkMagenta32', $FF8B008B ));
19086:     clDarkOliveGreen32', $FF556B2F ));
19087:     clDarkOrange32', $FFFF8C00 ));
19088:     clDarkOrchid32', $FF9932CC ));
19089:     clDarkRed32', $FF8B0000 ));
19090:     clDarkSalmon32', $FFE9967A ));
19091:     clDarkSeaGreen32', $FF8FBC8F ));
19092:     clDarkSlateBlue32', $FF483D8B ));
19093:     clDarkSlateGray32', $FF2F4F4F ));
19094:     clDarkSlateGrey32', $FF2F4F4F ));
19095:     clDarkTurquoise32', $FF00CED1 ));
19096:     clDarkViolet32', $FF9400D3 ));
19097:     clDeepPink32', $FFFF1493 ));
19098:     clDeepSkyBlue32', $FF00BFFF ));
19099:     clDodgerBlue32', $FF1E90FF ));
19100:     clFireBrick32', $FFB22222 ));
19101:     clFloralWhite32', $FFFFFAF0 ));
19102:     clGainsBoro32', $FFDCDCDC ));
19103:     clGhostWhite32', $FFF8F8FF ));
19104:     clGold32', $FFFFD700 ));
19105:     clGoldenRod32', $FFDAA520 ));
19106:     clGreenYellow32', $FFADFF2F ));
```

```
19107:     clGrey32', $FF808080 ));
19108:     clHoneyDew32', $FFF0FFF0 ));
19109:     clHotPink32', $FFFF69B4 ));
19110:     clIndianRed32', $FFCD5C5C ));
19111:     clIndigo32', $FF4B0082 ));
19112:     clIvory32', $FFFFFFF0 ));
19113:     clKhaki32', $FFF0E68C ));
19114:     clLavender32', $FFE6E6FA ));
19115:     clLavenderBlush32', $FFFFF0F5 ));
19116:     clLawnGreen32', $FF7CFC00 ));
19117:     clLemonChiffon32', $FFFFFACD ));
19118:     clLightBlue32', $FFADD8E6 ));
19119:     clLightCoral32', $FFF08080 ));
19120:     clLightCyan32', $FFE0FFFF ));
19121:     clLightGoldenRodYellow32', $FFFAFAD2 ));
19122:     clLightGreen32', $FF90EE90 ));
19123:     clLightGrey32', $FFD3D3D3 ));
19124:     clLightPink32', $FFFFB6C1 ));
19125:     clLightSalmon32', $FFFFA07A ));
19126:     clLightSeagreen32', $FF20B2AA ));
19127:     clLightSkyblue32', $FF87CEFA ));
19128:     clLightSlategray32', $FF778899 ));
19129:     clLightSlategrey32', $FF778899 ));
19130:     clLightSteelblue32', $FFB0C4DE ));
19131:     clLightYellow32', $FFFFFFE0 ));
19132:     clLtGray32', $FFC0C0C0 ));
19133:     clMedGray32', $FFA0A0A4 ));
19134:     clDkGray32', $FF808080 ));
19135:     clMoneyGreen32', $FFC0DCC0 ));
19136:     clLegacySkyBlue32', $FFA6CAF0 ));
19137:     clCream32', $FFFFFBF0 ));
19138:     clLimeGreen32', $FF32CD32 ));
19139:     clLinen32', $FFFAF0E6 ));
19140:     clMediumAquamarine32', $FF66CDAA ));
19141:     clMediumBlue32', $FF0000CD ));
19142:     clMediumOrchid32', $FFBA55D3 ));
19143:     clMediumPurple32', $FF9370DB ));
19144:     clMediumSeaGreen32', $FF3CB371 ));
19145:     clMediumSlateBlue32', $FF7B68EE ));
19146:     clMediumSpringGreen32', $FF00FA9A ));
19147:     clMediumTurquoise32', $FF48D1CC ));
19148:     clMediumVioletRed32', $FFC71585 ));
19149:     clMidnightBlue32', $FF191970 ));
19150:     clMintCream32', $FFF5FFFA ));
19151:     clMistyRose32', $FFFFE4E1 ));
19152:     clMoccasin32', $FFFFE4B5 ));
19153:     clNavajoWhite32', $FFFFDEAD ));
19154:     clOldLace32', $FFFDF5E6 ));
19155:     clOliveDrab32', $FF6B8E23 ));
19156:     clOrange32', $FFFFA500 ));
19157:     clOrangeRed32', $FFFF4500 ));
19158:     clOrchid32', $FFDA70D6 ));
19159:     clPaleGoldenRod32', $FFEEE8AA ));
19160:     clPaleGreen32', $FF98FB98 ));
19161:     clPaleTurquoise32', $FFAFEEEE ));
19162:     clPaleVioletred32', $FFDB7093 ));
19163:     clPapayaWhip32', $FFFFEFD5 ));
19164:     clPeachPuff32', $FFFFDAB9 ));
19165:     clPeru32', $FFCD853F ));
19166:     clPlum32', $FFDDA0DD ));
19167:     clPowderBlue32', $FFB0E0E6 ));
19168:     clRosyBrown32', $FFBC8F8F ));
19169:     clRoyalBlue32', $FF4169E1 ));
19170:     clSaddleBrown32', $FF8B4513 ));
19171:     clSalmon32', $FFFA8072 ));
19172:     clSandyBrown32', $FFF4A460 ));
19173:     clSeaGreen32', $FF2E8B57 ));
19174:     clSeaShell32', $FFFFF5EE ));
19175:     clSienna32', $FFA0522D ));
19176:     clSilver32', $FFC0C0C0 ));
19177:     clSkyblue32', $FF87CEEB ));
19178:     clSlateBlue32', $FF6A5ACD ));
19179:     clSlateGray32', $FF708090 ));
19180:     clSlateGrey32', $FF708090 ));
19181:     clSnow32', $FFFFFAFA ));
19182:     clSpringgreen32', $FF00FF7F ));
19183:     clSteelblue32', $FF4682B4 ));
19184:     clTan32', $FFD2B48C ));
19185:     clThistle32', $FFD8BFD8 ));
19186:     clTomato32', $FFFF6347 ));
19187:     clTurquoise32', $FF40E0D0 ));
19188:     clViolet32', $FFEE82EE ));
19189:     clWheat32', $FFF5DEB3 ));
19190:     clWhitesmoke32', $FFF5F5F5 ));
19191:     clYellowgreen32', $FF9ACD32 ));
19192:     clTrWhite32', $7FFFFFFF ));
19193:     clTrBlack32', $7F000000 ));
19194:     clTrRed32', $7FFF0000 ));
19195:     clTrGreen32', $7F00FF00 ));
```

```
19196:     clTrBlue32', $7F0000FF ));
19197:   // Fixed point math constants
19198:   FixedOne = $10000;  FixedHalf = $7FFF;
19199:   FixedPI  = Round(PI * FixedOne);
19200:   FixedToFloat = 1/FixedOne;
19201:
19202:  Special Types
19203: ************************************************
19204:   type Complex = record
19205:     X, Y : Float;
19206:   end;
19207:   type TVector     = array of Float;
19208:   TIntVector  = array of Integer;
19209:   TCompVector = array of Complex;
19210:   TBoolVector = array of Boolean;
19211:   TStrVector  = array of String;
19212:   TMatrix     = array of TVector;
19213:   TIntMatrix  = array of TIntVector;
19214:   TCompMatrix = array of TCompVector;
19215:   TBoolMatrix = array of TBoolVector;
19216:   TStrMatrix  = array of TStrVector;
19217:   TByteArray = array[0..32767] of byte; !
19218:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
19219:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
19220:   T2StringArray = array of array of string;
19221:   T2IntegerArray = array of array of integer;
19222:   AddTypeS('INT_PTR', 'Integer
19223:   AddTypeS('LONG_PTR', 'Integer
19224:   AddTypeS('UINT_PTR', 'Cardinal
19225:   AddTypeS('ULONG_PTR', 'Cardinal
19226:   AddTypeS('DWORD_PTR', 'ULONG_PTR
19227:   TIntegerDynArray', 'array of Integer
19228:   TCardinalDynArray', 'array of Cardinal
19229:   TWordDynArray', 'array of Word
19230:   TSmallIntDynArray', 'array of SmallInt
19231:   TByteDynArray', 'array of Byte
19232:   TShortIntDynArray', 'array of ShortInt
19233:   TInt64DynArray', 'array of Int64
19234:   TLongWordDynArray', 'array of LongWord
19235:   TSingleDynArray', 'array of Single
19236:   TDoubleDynArray', 'array of Double
19237:   TBooleanDynArray', 'array of Boolean
19238:   TStringDynArray', 'array of string
19239:   TWideStringDynArray', 'array of WideString
19240:   TDynByteArray     = array of Byte;
19241:   TDynShortintArray = array of Shortint;
19242:   TDynSmallintArray = array of Smallint;
19243:   TDynWordArray     = array of Word;
19244:   TDynIntegerArray  = array of Integer;
19245:   TDynLongintArray  = array of Longint;
19246:   TDynCardinalArray = array of Cardinal;
19247:   TDynInt64Array    = array of Int64;
19248:   TDynExtendedArray = array of Extended;
19249:   TDynDoubleArray   = array of Double;
19250:   TDynSingleArray   = array of Single;
19251:   TDynFloatArray    = array of Float;
19252:   TDynPointerArray  = array of Pointer;
19253:   TDynStringArray   = array of string;
19254:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
19255:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
19256:   TSynSearchOptions = set of TSynSearchOption;
19257:
19258:
19259: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
19260: ----------------------------------------------------------------------------
19261: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
19262: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
19263: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
19264: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19265: function CheckStringSum(vstring: string): integer;
19266: function HexToInt(HexNum: string): LongInt;
19267: function IntToBin(Int: Integer): String;
19268: function BinToInt(Binary: String): Integer;
19269: function HexToBin(HexNum: string): string; external2
19270: function BinToHex(Binary: String): string;
19271: function IntToFloat(i: Integer): double;
19272: function AddThousandSeparator(S: string; myChr: Char): string;
19273: function Max3(const X,Y,Z: Integer): Integer;
19274: procedure Swap(var X,Y: char); // faster without inline
19275: procedure ReverseString(var S: String);
19276: function CharToHexStr(Value: Char): string;
19277: function CharToUniCode(Value: Char): string;
19278: function Hex2Dec(Value: Str002): Byte;
19279: function HexStrCodeToStr(Value: string): string;
19280: function HexToStr(i: integer; value: string): string;
19281: function UniCodeToStr(Value: string): string;
19282: function CRC16(statement: string): string;
19283: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
19284: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
```

```
19285: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19286: Procedure ExecuteCommand(executeFile, paramstring: string);
19287: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
19288: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
19289: procedure SearchAndOpenDoc(vfilenamepath: string);
19290: procedure ShowInterfaces(myFile: string);
19291: function Fact2(av: integer): extended;
19292: Function BoolToStr(B: Boolean): string;
19293: Function GCD(x, y : LongInt) : LongInt;
19294: function LCM(m,n: longint): longint;
19295: function GetASCII: string;
19296: function GetItemHeight(Font: TFont): Integer;
19297: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
19298: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
19299: function getHINSTANCE: longword;
19300: function getHMODULE: longword;
19301: function GetASCII: string;
19302: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
19303: function WordIsOk(const AWord: string; var VW: Word): boolean;
19304: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
19305: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
19306: function SafeStr(const s: string): string;
19307: function ExtractUrlPath(const FileName: string): string;
19308: function ExtractUrlName(const FileName: string): string;
19309: function IsInternet: boolean;
19310: function RotateLeft1Bit_u32( Value: uint32): uint32;
19311: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats : Boolean);
19312: procedure getEnvironmentInfo;
19313: procedure AntiFreeze;
19314: function GetCPUSpeed: Double;
19315: function IsVirtualPcGuest : Boolean;
19316: function IsVmWareGuest : Boolean;
19317: procedure StartSerialDialog;
19318: function IsWoW64: boolean;
19319: function IsWow64String(var s: string): Boolean;
19320: procedure StartThreadDemo;
19321: Function RGB(R,G,B: Byte): TColor;
19322: Function Sendln(amess: string): boolean;
19323: Procedure maXbox;
19324: Function AspectRatio(aWidth, aHeight: Integer): String;
19325: function wget(aURL, afile: string): boolean;
19326: procedure PrintList(Value: TStringList);
19327: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
19328: procedure getEnvironmentInfo;
19329: procedure AntiFreeze;
19330: function getBitmap(apath: string): TBitmap;
19331: procedure ShowMessageBig(const aText : string);
19332: function YesNoDialog(const ACaption, AMsg: string): boolean;
19333: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
19334: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19335: //function myStrToBytes(const Value: String): TBytes;
19336: //function myBytesToStr(const Value: TBytes): String;
19337: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19338: function getBitmap(apath: string): TBitmap;
19339: procedure ShowMessageBig(const aText : string);
19340: Function StrToBytes(const Value: String): TBytes;
19341: Function BytesToStr(const Value: TBytes): String;
19342: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19343: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
       HostName:String):Bool;
19344: function FindInPaths(const fileName, paths : String) : String;
19345: procedure initHexArray(var hexn: THexArray);
19346: function josephusG(n,k: integer; var graphout: string): integer;
19347: function isPowerof2(num: int64): boolean;
19348: function powerOf2(exponent: integer): int64;
19349: function getBigPI: string;
19350: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
19351:  function GetASCIILine: string;
19352: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
19353:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
19354: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
19355: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
19356: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
19357: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
19358: function isKeypressed: boolean;
19359: function Keypress: boolean;
19360: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19361: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
19362: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
19363: function GetOSName: string;
19364: function GetOSVersion: string;
19365: function GetOSNumber: string;
19366: function getEnvironmentString: string;
19367: procedure StrReplace(var Str: String; Old, New: String);
19368: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19369: function getTeamViewerID: string;
19370: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
19371: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
```

```
19372: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19373: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
19374: function StartSocketService: Boolean;
19375: procedure StartSocketServiceForm;
19376: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
19377: function GetFileList1(apath: string): TStringlist;
19378: procedure LetFileList(FileList: TStringlist; apath: string);
19379: procedure StartWeb(aurl: string);
19380: function GetTodayFiles(startdir, amask: string): TStringlist;
19381: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
19382: function JavahashCode(val: string): Integer;
19383: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
19384: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
19385: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
19386: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
19387: Procedure ConvertToGray(Cnv: TCanvas);
19388: function GetFileDate(aFile:string; aWithTime:Boolean):string;
19389: procedure ShowMemory;
19390: function ShowMemory2: string;
19391: function getHostIP: string;
19392: procedure ShowBitmap(bmap: TBitmap);
19393: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
19394: function CreateDBGridForm(dblist: TStringList): TListbox;
19395: function isService: boolean;
19396: function isApplication: boolean;
19397: function isTerminalSession: boolean;
19398:
19399:
19400: // News of 3.9.8 up
19401: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19402: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19403: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19404: JvChart - TJvChart Component - 2009 Public
19405: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
19406: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
19407: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
19408: DMath DLL included incl. Demos
19409: Interface Navigator menu/View/Intf Navigator
19410: Unit Explorer menu/Debug/Units Explorer
19411: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
19412: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
19413: Script History to 9 Files WebServer light /Options/Addons/WebServer
19414: Full Text Finder, JVSimLogic Simulator Package
19415: Halt-Stop Program in Menu, WebServer2, Stop Event ,
19416: Conversion Routines, Prebuild Forms, CodeSearch
19417: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19418: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19419: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19420: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
19421: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
19422: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
19423: IDE Reflection API, Session Service Shell S3
19424: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
19425: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
19426: arduino map() function, PMRandom Generator
19427: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
19428: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
19429: REST Test Lib, Multilang Component, Forth Interpreter
19430: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
19431: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
19432: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19433: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19434: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
19435: QRCode Service, add more CFunctions like CDateTime of Synapse
19436: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
19437: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
19438: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
19439: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
19440: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
19441: BOLD Package, Indy Package5, maTRIx. MATHEMAX
19442: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
19443: emax layers: system-package-component-unit-class-function-block
19444: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
19445: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
19446: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
19447: OpenGL Game Demo: ..Options/Add Ons/Reversi
19448: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
19449: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
19450: 7% performance gain (hot spot profiling)
19451: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
19452: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
19453: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
19454: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
19455:
19456: add routines in 3.9.7.5
19457: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
19458: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
19459: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
19460: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
```

```
19461: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
19462: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
19463: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
19464:
19465: ////////////////////////// TestUnits //////////////////////////
19466:   SelftestPEM;
19467:   SelfTestCFundamentUtils;
19468:   SelfTestCFileUtils;
19469:   SelfTestCDateTime;
19470:   SelfTestCTimer;
19471:   SelfTestCRandom;
19472:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
19473:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
19474:
19475:   // Note: There's no need for installing a client certificate in the
19476:   //       webbrowser. The server asks the webbrowser to send a certificate but
19477:   //       if nothing is installed the software will work because the server
19478:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
19479:   //       file: c_cacert.p12
19480:   //       password: c_cakey
19481:
19482:   TGraphicControl = class(TControl)
19483:   private
19484:     FCanvas: TCanvas;
19485:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19486:   protected
19487:     procedure Paint; virtual;
19488:     property Canvas: TCanvas read FCanvas;
19489:   public
19490:     constructor Create(AOwner: TComponent); override;
19491:     destructor Destroy; override;
19492:   end;
19493:
19494:   TCustomControl = class(TWinControl)
19495:   private
19496:     FCanvas: TCanvas;
19497:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19498:   protected
19499:     procedure Paint; virtual;
19500:     procedure PaintWindow(DC: HDC); override;
19501:     property Canvas: TCanvas read FCanvas;
19502:   public
19503:     constructor Create(AOwner: TComponent); override;
19504:     destructor Destroy; override;
19505:   end;
19506:     RegisterPublishedProperties;
19507:     ('ONCHANGE', 'TNotifyEvent', iptrw);
19508:     ('ONCLICK', 'TNotifyEvent', iptrw);
19509:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
19510:     ('ONENTER', 'TNotifyEvent', iptrw);
19511:     ('ONEXIT', 'TNotifyEvent', iptrw);
19512:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
19513:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
19514:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
19515:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
19516:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
19517: //*************************************************************************
19518:   // To stop the while loop, click on Options/Show Include (boolean switch)!
19519:   Control a loop in a script with a form event:
19520:   IncludeON;   //control the while loop
19521:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
19522:
19523: //-------------------------------------------------------------------------
19524: //*************mX4 ini-file Configuration**********************************
19525: //-------------------------------------------------------------------------
19526:   using config file maxboxdef.ini       menu/Help/Config File
19527:
19528: //*** Definitions for maXbox mX3 ***
19529: [FORM]
19530: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
19531: FONTSIZE=14
19532: EXTENSION=txt
19533: SCREENX=1386
19534: SCREENY=1077
19535: MEMHEIGHT=350
19536: PRINTFONT=Courier New //GUI Settings
19537: LINENUMBERS=Y   //line numbers at gutter in editor at left side
19538: EXCEPTIONLOG=Y  //store excepts and success in 2 log files see below! -menu Debug/Show Last Exceptions
19539: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
19540: BOOTSCRIPT=Y    //enabling load a boot script
19541: MEMORYREPORT=Y  //shows memory report on closing maXbox
19542: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
19543: NAVIGATOR=N     //shows function list at the right side of editor
19544: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
19545: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
19546: [WEB]
19547: IPPORT=8080     //for internal webserver - menu /Options/Add Ons/WebServer
19548: IPHOST=192.168.1.53
19549: ROOTCERT=filepathY
```

```
19550: SCERT=filepathY
19551: RSAKEY=filepathY
19552: VERSIONCHECK=Y
19553:
19554: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
19555:
19556: Also possible to set report memory in script to override ini setting
19557: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19558:
19559:
19560: After Change the ini file you can reload the file with ../Help/Config Update
19561:
19562: //-------------------------------------------------------------------------
19563: //**************mX4 maildef.ini ini-file Configuration********************
19564: //-------------------------------------------------------------------------
19565: //*** Definitions for maXMail ***
19566: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
19567: [MAXMAIL]
19568: HOST=getmail.softwareschule.ch
19569: USER=mailusername
19570: PASS=password
19571: PORT=110
19572: SSL=Y
19573: BODY=Y
19574: LAST=5
19575:
19576: ADO Connection String:
19577: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
19578:
19579: \452_dbtreeview2access.txt
19580: program TestDbTreeViewMainForm2_ACCESS;
19581:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
19582:                    +Exepath+'\examples\detail.mdb;Persist Security Info=False';
19583:
19584: OpenSSL Lib:  unit ssl_openssl_lib;
19585: {$IFDEF CIL}
19586: const
19587:   {$IFDEF LINUX}
19588:   DLLSSLName = 'libssl.so';
19589:   DLLUtilName = 'libcrypto.so';
19590:   {$ELSE}
19591:   DLLSSLName = 'ssleay32.dll';
19592:   DLLUtilName = 'libeay32.dll';
19593:   {$ENDIF}
19594: {$ELSE}
19595: var
19596:   {$IFNDEF MSWINDOWS}
19597:     {$IFDEF DARWIN}
19598:     DLLSSLName: string = 'libssl.dylib';
19599:     DLLUtilName: string = 'libcrypto.dylib';
19600:     {$ELSE}
19601:     DLLSSLName: string = 'libssl.so';
19602:     DLLUtilName: string = 'libcrypto.so';
19603:     {$ENDIF}
19604:   {$ELSE}
19605:   DLLSSLName: string = 'ssleay32.dll';
19606:   DLLSSLName2: string = 'libssl32.dll';
19607:   DLLUtilName: string = 'libeay32.dll';
19608:   {$ENDIF}
19609: {$ENDIF}
19610:
19611:
19612: //-------------------------------------------------------------------------
19613: //**************mX4 Macro Tags  ********************************************
19614: //---------------------------------------------------------- ----------------
19615:
19616:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
       E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
19617:
19618: //Tag Macros
19619:
19620:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
19621:
19622: //Tag Macros
19623: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
19624: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
19625: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
19626: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
19627: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
19628: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
19629: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
19630: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
19631: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
19632:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
19633: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
19634: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
19635:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
19636: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
19637:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
```

```
19638: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
19639:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
19640:
19641: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
19642:
19643: //Replace Macros
19644:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
19645:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
19646:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
19647:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
19648:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
19649:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
19650:
19651:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
19652:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
19653: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
19654:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
19655:
19656: //-------------------------------------------------------------------------
19657: //*************mX4 ToDo List Tags  ../Help/ToDo List************************
19658: //-------------------------------------------------------- ---------------------
19659:
19660:     while I < sl.Count do begin
19661: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
19662:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
19663:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
19664:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
19665:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
19666:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
19667:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
19668:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
19669:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
19670:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
19671:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
19672:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
19673:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
19674:       Inc(I);
19675:     end;
19676:
19677:
19678: //-------------------------------------------------------------------------
19679: //*************mX4 Public  Tools API ****************************************
19680: //-------------------------------------------------------------------------------
19681:   file : unit uPSI_fMain.pas;               {$OTAP} Open Tools API Catalog
19682:  // Those functions concern the editor and preprocessor, all of the IDE
19683:  Example: Call it with maxform1.Info1Click(self)
19684:  Note: Call all Methods with maxForm1., e.g.:
19685:                        maxForm1.ShellStyle1Click(self);
19686:
19687: procedure SIRegister_fMain(CL: TPSPascalCompiler);
19688: begin
19689:  Const('BYTECODE','String 'bytecode.txt'
19690:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
19691:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
19692:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
19693:  Const('PSINC','String PS Includes (*.inc)|*.INC
19694:  Const('DEFFILENAME','String 'firstdemo.txt
19695:  Const('DEFINIFILE','String 'maxboxdef.ini
19696:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
19697:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
19698:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
19699:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
19700:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
19701:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
19702:  Const('INCLUDEBOX','String 'pas_includebox.inc
19703:  Const('BOOTSCRIPT','String 'maxbootscript.txt
19704:  Const('MBVERSION','String '3.9.9.98
19705:  Const('VERSION','String'3.9.9.98
19706:  Const('MBVER','String '399
19707:  Const('MBVERI','Integer'(399);
19708:  Const('MBVERIALL','Integer'(39998);
19709:  Const('EXENAME','String 'maXbox3.exe
19710:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
19711:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
19712:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
19713:  Const('MXINTERNETCHECK','String 'www.ask.com
19714:  Const('MXMAIL','String 'max@kleiner.com
19715:  Const('TAB','Char #$09);
19716:  Const('CODECOMPLETION','String 'bds_delphi.dci
19717:  SIRegister_TMaxForm1(CL);
19718: end;
19719:
19720:   with FindClass('TForm'),'TMaxForm1') do begin
19721:     memo2', 'TMemo', iptrw);
19722:     memo1', 'TSynMemo', iptrw);
19723:     CB1SCList', 'TComboBox', iptrw);
19724:     mxNavigator', 'TComboBox', iptrw);
19725:     IPHost', 'string', iptrw);
19726:     IPPort', 'integer', iptrw);
```

```
19727:     COMPort', 'integer', iptrw);      //3.9.6.4
19728:     Splitter1', 'TSplitter', iptrw);
19729:     PSScript', 'TPSScript', iptrw);
19730:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
19731:     MainMenu1', 'TMainMenu', iptrw);
19732:     Program1', 'TMenuItem', iptrw);
19733:     Compile1', 'TMenuItem', iptrw);
19734:     Files1', 'TMenuItem', iptrw);
19735:     open1', 'TMenuItem', iptrw);
19736:     Save2', 'TMenuItem', iptrw);
19737:     Options1', 'TMenuItem', iptrw);
19738:     Savebefore1', 'TMenuItem', iptrw);
19739:     Largefont1', 'TMenuItem', iptrw);
19740:     sBytecode1', 'TMenuItem', iptrw);
19741:     Saveas3', 'TMenuItem', iptrw);
19742:     Clear1', 'TMenuItem', iptrw);
19743:     Slinenumbers1', 'TMenuItem', iptrw);
19744:     About1', 'TMenuItem', iptrw);
19745:     Search1', 'TMenuItem', iptrw);
19746:     SynPasSyn1', 'TSynPasSyn', iptrw);
19747:     memo1', 'TSynMemo', iptrw);
19748:     SynEditSearch1', 'TSynEditSearch', iptrw);
19749:     WordWrap1', 'TMenuItem', iptrw);
19750:     XPManifest1', 'TXPManifest', iptrw);
19751:     SearchNext1', 'TMenuItem', iptrw);
19752:     Replace1', 'TMenuItem', iptrw);
19753:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
19754:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
19755:     ShowInclude1', 'TMenuItem', iptrw);
19756:     SynEditPrint1', 'TSynEditPrint', iptrw);
19757:     Printout1', 'TMenuItem', iptrw);
19758:     mnPrintColors1', 'TMenuItem', iptrw);
19759:     dlgFilePrint', 'TPrintDialog', iptrw);
19760:     dlgPrintFont1', 'TFontDialog', iptrw);
19761:     mnuPrintFont1', 'TMenuItem', iptrw);
19762:     Include1', 'TMenuItem', iptrw);
19763:     CodeCompletionList1', 'TMenuItem', iptrw);
19764:     IncludeList1', 'TMenuItem', iptrw);
19765:     ImageList1', 'TImageList', iptrw);
19766:     ImageList2', 'TImageList', iptrw);
19767:     CoolBar1', 'TCoolBar', iptrw);
19768:     ToolBar1', 'TToolBar', iptrw);
19769:     tbtnLoad', 'TToolButton', iptrw);
19770:     ToolButton2', 'TToolButton', iptrw);
19771:     tbtnFind', 'TToolButton', iptrw);
19772:     tbtnCompile', 'TToolButton', iptrw);
19773:     tbtnTrans', 'TToolButton', iptrw);
19774:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
19775:     toolbtnTutorial', 'TToolButton', iptrw);
19776:     tbtn6res', 'TToolButton', iptrw);
19777:     ToolButton5', 'TToolButton', iptrw);
19778:     ToolButton1', 'TToolButton', iptrw);
19779:     ToolButton3', 'TToolButton', iptrw);
19780:     statusBar1', 'TStatusBar', iptrw);
19781:     SaveOutput1', 'TMenuItem', iptrw);
19782:     ExportClipboard1', 'TMenuItem', iptrw);
19783:     Close1', 'TMenuItem', iptrw);
19784:     Manual1', 'TMenuItem', iptrw);
19785:     About2', 'TMenuItem', iptrw);
19786:     loadLastfile1', 'TMenuItem', iptrw);
19787:     imglogo', 'TImage', iptrw);
19788:     cedebug', 'TPSScriptDebugger', iptrw);
19789:     debugPopupMenu1', 'TPopupMenu', iptrw);
19790:     BreakPointMenu', 'TMenuItem', iptrw);
19791:     Decompile1', 'TMenuItem', iptrw);
19792:     StepInto1', 'TMenuItem', iptrw);
19793:     StepOut1', 'TMenuItem', iptrw);
19794:     Reset1', 'TMenuItem', iptrw);
19795:     DebugRun1', 'TMenuItem', iptrw);
19796:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
19797:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
19798:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
19799:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
19800:     tutorial4', 'TMenuItem', iptrw);
19801:     ExporttoClipboard1', 'TMenuItem', iptrw);
19802:     ImportfromClipboard1', 'TMenuItem', iptrw);
19803:     N4', 'TMenuItem', iptrw);
19804:     N5', 'TMenuItem', iptrw);
19805:     N6', 'TMenuItem', iptrw);
19806:     ImportfromClipboard2', 'TMenuItem', iptrw);
19807:     tutorial1', 'TMenuItem', iptrw);
19808:     N7', 'TMenuItem', iptrw);
19809:     ShowSpecChars1', 'TMenuItem', iptrw);
19810:     OpenDirectory1', 'TMenuItem', iptrw);
19811:     procMess', 'TMenuItem', iptrw);
19812:     tbtnUseCase', 'TToolButton', iptrw);
19813:     ToolButton7', 'TToolButton', iptrw);
19814:     EditFont1', 'TMenuItem', iptrw);
19815:     UseCase1', 'TMenuItem', iptrw);
```

```
19816:     tutorial21', 'TMenuItem', iptrw);
19817:     OpenUseCase1', 'TMenuItem', iptrw);
19818:     PSImport_DB1', 'TPSImport_DB', iptrw);
19819:     tutorial31', 'TMenuItem', iptrw);
19820:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
19821:     HTMLSyntax1', 'TMenuItem', iptrw);
19822:     ShowInterfaces1', 'TMenuItem', iptrw);
19823:     Tutorial5', 'TMenuItem', iptrw);
19824:     AllFunctionsList1', 'TMenuItem', iptrw);
19825:     ShowLastException1', 'TMenuItem', iptrw);
19826:     PlayMP31', 'TMenuItem', iptrw);
19827:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
19828:     texSyntax1', 'TMenuItem', iptrw);
19829:     N8', 'TMenuItem', iptrw);
19830:     GetEMails1', 'TMenuItem', iptrw);
19831:     SynCppSyn1', 'TSynCppSyn', iptrw);
19832:     CSyntax1', 'TMenuItem', iptrw);
19833:     Tutorial6', 'TMenuItem', iptrw);
19834:     New1', 'TMenuItem', iptrw);
19835:     AllObjectsList1', 'TMenuItem', iptrw);
19836:     LoadBytecode1', 'TMenuItem', iptrw);
19837:     CipherFile1', 'TMenuItem', iptrw);
19838:     N9', 'TMenuItem', iptrw);
19839:     N10', 'TMenuItem', iptrw);
19840:     Tutorial11', 'TMenuItem', iptrw);
19841:     Tutorial71', 'TMenuItem', iptrw);
19842:     UpdateService1', 'TMenuItem', iptrw);
19843:     PascalSchool1', 'TMenuItem', iptrw);
19844:     Tutorial81', 'TMenuItem', iptrw);
19845:     DelphiSite1', 'TMenuItem', iptrw);
19846:     Output1', 'TMenuItem', iptrw);
19847:     TerminalStyle1', 'TMenuItem', iptrw);
19848:     ReadOnly1', 'TMenuItem', iptrw);
19849:     ShellStyle1', 'TMenuItem', iptrw);
19850:     BigScreen1', 'TMenuItem', iptrw);
19851:     Tutorial91', 'TMenuItem', iptrw);
19852:     SaveOutput2', 'TMenuItem', iptrw);
19853:     N11', 'TMenuItem', iptrw);
19854:     SaveScreenshot', 'TMenuItem', iptrw);
19855:     Tutorial101', 'TMenuItem', iptrw);
19856:     SQLSyntax1', 'TMenuItem', iptrw);
19857:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
19858:     Console1', 'TMenuItem', iptrw);
19859:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
19860:     XMLSyntax1', 'TMenuItem', iptrw);
19861:     ComponentCount1', 'TMenuItem', iptrw);
19862:     NewInstance1', 'TMenuItem', iptrw);
19863:     toolbtnTutorial', 'TToolButton', iptrw);
19864:     Memory1', 'TMenuItem', iptrw);
19865:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
19866:     JavaSyntax1', 'TMenuItem', iptrw);
19867:     SyntaxCheck1', 'TMenuItem', iptrw);
19868:     Tutorial10Statistics1', 'TMenuItem', iptrw);
19869:     ScriptExplorer1', 'TMenuItem', iptrw);
19870:     FormOutput1', 'TMenuItem', iptrw);
19871:     ArduinoDump1', 'TMenuItem', iptrw);
19872:     AndroidDump1', 'TMenuItem', iptrw);
19873:     GotoEnd1', 'TMenuItem', iptrw);
19874:     AllResourceList1', 'TMenuItem', iptrw);
19875:     ToolButton4', 'TToolButton', iptrw);
19876:     tbtn6res', 'TToolButton', iptrw);
19877:     Tutorial11Forms1', 'TMenuItem', iptrw);
19878:     Tutorial12SQL1', 'TMenuItem', iptrw);
19879:     ResourceExplore1', 'TMenuItem', iptrw);
19880:     Info1', 'TMenuItem', iptrw);
19881:     N12', 'TMenuItem', iptrw);
19882:     CryptoBox1', 'TMenuItem', iptrw);
19883:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
19884:     CipherFile2', 'TMenuItem', iptrw);
19885:     N13', 'TMenuItem', iptrw);
19886:     ModulesCount1', 'TMenuItem', iptrw);
19887:     AddOns2', 'TMenuItem', iptrw);
19888:     N4GewinntGame1', 'TMenuItem', iptrw);
19889:     DocuforAddOns1', 'TMenuItem', iptrw);
19890:     Tutorial14Async1', 'TMenuItem', iptrw);
19891:     Lessons15Review1', 'TMenuItem', iptrw);
19892:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
19893:     PHPSyntax1', 'TMenuItem', iptrw);
19894:     Breakpoint1', 'TMenuItem', iptrw);
19895:     SerialRS2321', 'TMenuItem', iptrw);
19896:     N14', 'TMenuItem', iptrw);
19897:     SynCSSyn1', 'TSynCSSyn', iptrw);
19898:     CSyntax2', 'TMenuItem', iptrw);
19899:     Calculator1', 'TMenuItem', iptrw);
19900:     tbtnSerial', 'TToolButton', iptrw);
19901:     ToolButton8', 'TToolButton', iptrw);
19902:     Tutorial151', 'TMenuItem', iptrw);
19903:     N15', 'TMenuItem', iptrw);
19904:     N16', 'TMenuItem', iptrw);
```

```
19905:     ControlBar1', 'TControlBar', iptrw);
19906:     ToolBar2', 'TToolBar', iptrw);
19907:     BtnOpen', 'TToolButton', iptrw);
19908:     BtnSave', 'TToolButton', iptrw);
19909:     BtnPrint', 'TToolButton', iptrw);
19910:     BtnColors', 'TToolButton', iptrw);
19911:     btnClassReport', 'TToolButton', iptrw);
19912:     BtnRotateRight', 'TToolButton', iptrw);
19913:     BtnFullSize', 'TToolButton', iptrw);
19914:     BtnFitToWindowSize', 'TToolButton', iptrw);
19915:     BtnZoomMinus', 'TToolButton', iptrw);
19916:     BtnZoomPlus', 'TToolButton', iptrw);
19917:     Panel1', 'TPanel', iptrw);
19918:     LabelBrettgroesse', 'TLabel', iptrw);
19919:     CB1SCList', 'TComboBox', iptrw);
19920:     ImageListNormal', 'TImageList', iptrw);
19921:     spbtnexplore', 'TSpeedButton', iptrw);
19922:     spbtnexample', 'TSpeedButton', iptrw);
19923:     spbsaveas', 'TSpeedButton', iptrw);
19924:     imglogobox', 'TImage', iptrw);
19925:     EnlargeFont1', 'TMenuItem', iptrw);
19926:     EnlargeFont2', 'TMenuItem', iptrw);
19927:     ShrinkFont1', 'TMenuItem', iptrw);
19928:     ThreadDemo1', 'TMenuItem', iptrw);
19929:     HEXEditor1', 'TMenuItem', iptrw);
19930:     HEXView1', 'TMenuItem', iptrw);
19931:     HEXInspect1', 'TMenuItem', iptrw);
19932:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
19933:     ExporttoHTML1', 'TMenuItem', iptrw);
19934:     ClassCount1', 'TMenuItem', iptrw);
19935:     HTMLOutput1', 'TMenuItem', iptrw);
19936:     HEXEditor2', 'TMenuItem', iptrw);
19937:     Minesweeper1', 'TMenuItem', iptrw);
19938:     N17', 'TMenuItem', iptrw);
19939:     PicturePuzzle1', 'TMenuItem', iptrw);
19940:     sbvclhelp', 'TSpeedButton', iptrw);
19941:     DependencyWalker1', 'TMenuItem', iptrw);
19942:     WebScanner1', 'TMenuItem', iptrw);
19943:     View1', 'TMenuItem', iptrw);
19944:     mnToolbar1', 'TMenuItem', iptrw);
19945:     mnStatusbar2', 'TMenuItem', iptrw);
19946:     mnConsole2', 'TMenuItem', iptrw);
19947:     mnCoolbar2', 'TMenuItem', iptrw);
19948:     mnSplitter2', 'TMenuItem', iptrw);
19949:     WebServer1', 'TMenuItem', iptrw);
19950:     Tutorial17Server1', 'TMenuItem', iptrw);
19951:     Tutorial18Arduino1', 'TMenuItem', iptrw);
19952:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
19953:     PerlSyntax1', 'TMenuItem', iptrw);
19954:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
19955:     PythonSyntax1', 'TMenuItem', iptrw);
19956:     DMathLibrary1', 'TMenuItem', iptrw);
19957:     IntfNavigator1', 'TMenuItem', iptrw);
19958:     EnlargeFontConsole1', 'TMenuItem', iptrw);
19959:     ShrinkFontConsole1', 'TMenuItem', iptrw);
19960:     SetInterfaceList1', 'TMenuItem', iptrw);
19961:     popintfList', 'TPopupMenu', iptrw);
19962:     intfAdd1', 'TMenuItem', iptrw);
19963:     intfDelete1', 'TMenuItem', iptrw);
19964:     intfRefactor1', 'TMenuItem', iptrw);
19965:     Defactor1', 'TMenuItem', iptrw);
19966:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
19967:     Tutorial20Regex', 'TMenuItem', iptrw);
19968:     N18', 'TMenuItem', iptrw);
19969:     ManualE1', 'TMenuItem', iptrw);
19970:     FullTextFinder1', 'TMenuItem', iptrw);
19971:     Move1', 'TMenuItem', iptrw);
19972:     FractalDemo1', 'TMenuItem', iptrw);
19973:     Tutorial21Android1', 'TMenuItem', iptrw);
19974:     Tutorial0Function1', 'TMenuItem', iptrw);
19975:     SimuLogBox1', 'TMenuItem', iptrw);
19976:     OpenExamples1', 'TMenuItem', iptrw);
19977:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
19978:     JavaScriptSyntax1', 'TMenuItem', iptrw);
19979:     Halt1', 'TMenuItem', iptrw);
19980:     CodeSearch1', 'TMenuItem', iptrw);
19981:     SynRubySyn1', 'TSynRubySyn', iptrw);
19982:     RubySyntax1', 'TMenuItem', iptrw);
19983:     Undo1', 'TMenuItem', iptrw);
19984:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
19985:     LinuxShellScript1', 'TMenuItem', iptrw);
19986:     Rename1', 'TMenuItem', iptrw);
19987:     spdcodesearch', 'TSpeedButton', iptrw);
19988:     Preview1', 'TMenuItem', iptrw);
19989:     Tutorial22Services1', 'TMenuItem', iptrw);
19990:     Tutorial23RealTime1', 'TMenuItem', iptrw);
19991:     Configuration1', 'TMenuItem', iptrw);
19992:     MP3Player1', 'TMenuItem', iptrw);
19993:     DLLSpy1', 'TMenuItem', iptrw);
```

```
19994:     SynURIOpener1', 'TSynURIOpener', iptrw);
19995:     SynURISyn1', 'TSynURISyn', iptrw);
19996:     URILinksClicks1', 'TMenuItem', iptrw);
19997:     EditReplace1', 'TMenuItem', iptrw);
19998:     GotoLine1', 'TMenuItem', iptrw);
19999:     ActiveLineColor1', 'TMenuItem', iptrw);
20000:     ConfigFile1', 'TMenuItem', iptrw);
20001:     Sort1Intflist', 'TMenuItem', iptrw);
20002:     Redo1', 'TMenuItem', iptrw);
20003:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
20004:     Tutorial25Configuration1', 'TMenuItem', iptrw);
20005:     IndentSelection1', 'TMenuItem', iptrw);
20006:     UnindentSection1', 'TMenuItem', iptrw);
20007:     SkyStyle1', 'TMenuItem', iptrw);
20008:     N19', 'TMenuItem', iptrw);
20009:     CountWords1', 'TMenuItem', iptrw);
20010:     imbookmarkimages', 'TImageList', iptrw);
20011:     Bookmark11', 'TMenuItem', iptrw);
20012:     N20', 'TMenuItem', iptrw);
20013:     Bookmark21', 'TMenuItem', iptrw);
20014:     Bookmark31', 'TMenuItem', iptrw);
20015:     Bookmark41', 'TMenuItem', iptrw);
20016:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
20017:
20018:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
20019:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
20020:     Procedure PSScriptCompile( Sender : TPSScript)
20021:     Procedure Compile1Click( Sender : TObject)
20022:     Procedure PSScriptExecute( Sender : TPSScript)
20023:     Procedure open1Click( Sender : TObject)
20024:     Procedure Save2Click( Sender : TObject)
20025:     Procedure Savebefore1Click( Sender : TObject)
20026:     Procedure Largefont1Click( Sender : TObject)
20027:     Procedure FormActivate( Sender : TObject)
20028:     Procedure SBytecode1Click( Sender : TObject)
20029:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
20030:     Procedure Saveas3Click( Sender : TObject)
20031:     Procedure Clear1Click( Sender : TObject)
20032:     Procedure Slinenumbers1Click( Sender : TObject)
20033:     Procedure About1Click( Sender : TObject)
20034:     Procedure Search1Click( Sender : TObject)
20035:     Procedure FormCreate( Sender : TObject)
20036:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
20037:                                                     var Action : TSynReplaceAction)
20038:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
20039:     Procedure WordWrap1Click( Sender : TObject)
20040:     Procedure SearchNext1Click( Sender : TObject)
20041:     Procedure Replace1Click( Sender : TObject)
20042:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
20043:     Procedure ShowInclude1Click( Sender : TObject)
20044:     Procedure Printout1Click( Sender : TObject)
20045:     Procedure mnuPrintFont1Click( Sender : TObject)
20046:     Procedure Include1Click( Sender : TObject)
20047:     Procedure FormDestroy( Sender : TObject)
20048:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
20049:     Procedure UpdateView1Click( Sender : TObject)
20050:     Procedure CodeCompletionList1Click( Sender : TObject)
20051:     Procedure SaveOutput1Click( Sender : TObject)
20052:     Procedure ExportClipboard1Click( Sender : TObject)
20053:     Procedure Close1Click( Sender : TObject)
20054:     Procedure Manual1Click( Sender : TObject)
20055:     Procedure LoadLastFile1Click( Sender : TObject)
20056:     Procedure Memo1Change( Sender : TObject)
20057:     Procedure Decompile1Click( Sender : TObject)
20058:     Procedure StepInto1Click( Sender : TObject)
20059:     Procedure StepOut1Click( Sender : TObject)
20060:     Procedure Reset1Click( Sender : TObject)
20061:     Procedure cedebugAfterExecute( Sender : TPSScript)
20062:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
20063:     Procedure cedebugCompile( Sender : TPSScript)
20064:     Procedure cedebugExecute( Sender : TPSScript)
20065:     Procedure cedebugIdle( Sender : TObject)
20066:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
20067:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
20068:     Procedure BreakPointMenuClick( Sender : TObject)
20069:     Procedure DebugRun1Click( Sender : TObject)
20070:     Procedure tutorial4Click( Sender : TObject)
20071:     Procedure ImportfromClipboard1Click( Sender : TObject)
20072:     Procedure ImportfromClipboard2Click( Sender : TObject)
20073:     Procedure tutorial1Click( Sender : TObject)
20074:     Procedure ShowSpecChars1Click( Sender : TObject)
20075:     Procedure StatusBar1DblClick( Sender : TObject)
20076:     Procedure PSScriptLine( Sender : TObject)
20077:     Procedure OpenDirectory1Click( Sender : TObject)
20078:     Procedure procMessClick( Sender : TObject)
20079:     Procedure tbtnUseCaseClick( Sender : TObject)
20080:     Procedure EditFont1Click( Sender : TObject)
20081:     Procedure tutorial21Click( Sender : TObject)
20082:     Procedure tutorial31Click( Sender : TObject)
```

```
20083:     Procedure HTMLSyntax1Click( Sender : TObject)
20084:     Procedure ShowInterfaces1Click( Sender : TObject)
20085:     Procedure Tutorial5Click( Sender : TObject)
20086:     Procedure ShowLastException1Click( Sender : TObject)
20087:     Procedure PlayMP31Click( Sender : TObject)
20088:     Procedure AllFunctionsList1Click( Sender : TObject)
20089:     Procedure texSyntax1Click( Sender : TObject)
20090:     Procedure GetEMails1Click( Sender : TObject)
20091:     procedure DelphiSite1Click(Sender: TObject);
20092:     procedure TerminalStyle1Click(Sender: TObject);
20093:     procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
20094:     procedure ShellStyle1Click(Sender: TObject);
20095:     procedure Console1Click(Sender: TObject);     //3.2
20096:     procedure BigScreen1Click(Sender: TObject);
20097:     procedure Tutorial91Click(Sender: TObject);
20098:     procedure SaveScreenshotClick(Sender: TObject);
20099:     procedure Tutorial101Click(Sender: TObject);
20100:     procedure SQLSyntax1Click(Sender: TObject);
20101:     procedure XMLSyntax1Click(Sender: TObject);
20102:     procedure ComponentCount1Click(Sender: TObject);
20103:     procedure NewInstance1Click(Sender: TObject);
20104:     procedure CSyntax1Click(Sender: TObject);
20105:     procedure Tutorial6Click(Sender: TObject);
20106:     procedure New1Click(Sender: TObject);
20107:     procedure AllObjectsList1Click(Sender: TObject);
20108:     procedure LoadBytecode1Click(Sender: TObject);
20109:     procedure CipherFile1Click(Sender: TObject);  //V3.5
20110:     procedure NewInstance1Click(Sender: TObject);
20111:     procedure toolbtnTutorialClick(Sender: TObject);
20112:     procedure Memory1Click(Sender: TObject);
20113:     procedure JavaSyntax1Click(Sender: TObject);
20114:     procedure SyntaxCheck1Click(Sender: TObject);
20115:     procedure ScriptExplorer1Click(Sender: TObject);
20116:     procedure FormOutput1Click(Sender: TObject);  //V3.6
20117:     procedure GotoEnd1Click(Sender: TObject);
20118:     procedure AllResourceList1Click(Sender: TObject);
20119:     procedure tbtn6resClick(Sender: TObject);   //V3.7
20120:     procedure Info1Click(Sender: TObject);
20121:     procedure Tutorial10Statistics1Click(Sender: TObject);
20122:     procedure Tutorial11Forms1Click(Sender: TObject);
20123:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
20124:     procedure ResourceExplore1Click(Sender: TObject);
20125:     procedure Info1Click(Sender: TObject);
20126:     procedure CryptoBox1Click(Sender: TObject);
20127:     procedure ModulesCount1Click(Sender: TObject);
20128:     procedure N4GewinntGame1Click(Sender: TObject);
20129:     procedure PHPSyntax1Click(Sender: TObject);
20130:     procedure SerialRS2321Click(Sender: TObject);
20131:     procedure CSyntax2Click(Sender: TObject);
20132:     procedure Calculator1Click(Sender: TObject);
20133:     procedure Tutorial13Ciphering1Click(Sender: TObject);
20134:     procedure Tutorial14Async1Click(Sender: TObject);
20135:     procedure PHPSyntax1Click(Sender: TObject);
20136:     procedure BtnZoomPlusClick(Sender: TObject);
20137:     procedure BtnZoomMinusClick(Sender: TObject);
20138:     procedure btnClassReportClick(Sender: TObject);
20139:     procedure ThreadDemo1Click(Sender: TObject);
20140:     procedure HEXView1Click(Sender: TObject);
20141:     procedure ExporttoHTML1Click(Sender: TObject);
20142:     procedure Minesweeper1Click(Sender: TObject);
20143:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
20144:     procedure sbvclhelpClick(Sender: TObject);
20145:     procedure DependencyWalker1Click(Sender: TObject);
20146:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
20147:     procedure WebScanner1Click(Sender: TObject);
20148:     procedure mnToolbar1Click(Sender: TObject);
20149:     procedure mnStatusbar2Click(Sender: TObject);
20150:     procedure mnConsole2Click(Sender: TObject);
20151:     procedure mnCoolbar2Click(Sender: TObject);
20152:     procedure mnSplitter2Click(Sender: TObject);
20153:     procedure WebServer1Click(Sender: TObject);
20154:     procedure PerlSyntax1Click(Sender: TObject);
20155:     procedure PythonSyntax1Click(Sender: TObject);
20156:     procedure DMathLibrary1Click(Sender: TObject);
20157:     procedure IntfNavigator1Click(Sender: TObject);
20158:     procedure FullTextFinder1Click(Sender: TObject);
20159:     function AppName: string;
20160:     function ScriptName: string;
20161:     function LastName: string;
20162:     procedure FractalDemo1Click(Sender: TObject);
20163:     procedure SimuLogBox1Click(Sender: TObject);
20164:     procedure OpenExamples1Click(Sender: TObject);
20165:     procedure Halt1Click(Sender: TObject);
20166:     procedure Stop;
20167:     procedure CodeSearch1Click(Sender: TObject);
20168:     procedure RubySyntax1Click(Sender: TObject);
20169:     procedure Undo1Click(Sender: TObject);
20170:     procedure LinuxShellScript1Click(Sender: TObject);
20171:     procedure WebScannerDirect(urls: string);
```

```
20172:     procedure WebScanner(urls: string);
20173:     procedure LoadInterfaceList2;
20174:     procedure DLLSpy1Click(Sender: TObject);
20175:     procedure Memo1DblClick(Sender: TObject);
20176:     procedure URILinksClicks1Click(Sender: TObject);
20177:     procedure GotoLine1Click(Sender: TObject);
20178:     procedure ConfigFile1Click(Sender: TObject);
20179:     Procedure Sort1IntflistClick( Sender : TObject)
20180:     Procedure Redo1Click( Sender : TObject)
20181:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
20182:     Procedure IndentSelection1Click( Sender : TObject)
20183:     Procedure UnindentSection1Click( Sender : TObject)
20184:     Procedure SkyStyle1Click( Sender : TObject)
20185:     Procedure CountWords1Click( Sender : TObject)
20186:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
20187:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
20188:     Procedure Bookmark11Click( Sender : TObject)
20189:     Procedure Bookmark21Click( Sender : TObject)
20190:     Procedure Bookmark31Click( Sender : TObject)
20191:     Procedure Bookmark41Click( Sender : TObject)
20192:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
20193:     'STATMemoryReport', 'boolean', iptrw);
20194:     'IPPort', 'integer', iptrw);
20195:     'COMPort', 'integer', iptrw);
20196:     'lbintflist', 'TListBox', iptrw);
20197:     Function GetStatChange : boolean
20198:     Procedure SetStatChange( vstat : boolean)
20199:     Function GetActFileName : string
20200:     Procedure SetActFileName( vname : string)
20201:     Function GetLastFileName : string
20202:     Procedure SetLastFileName( vname : string)
20203:     Procedure WebScannerDirect( urls : string)
20204:     Procedure LoadInterfaceList2
20205:     Function GetStatExecuteShell : boolean
20206:     Procedure DoEditorExecuteCommand( EditorCommand : word)
20207:     function GetActiveLineColor: TColor
20208:     procedure SetActiveLineColor(acolor: TColor)
20209:     procedure ScriptListbox1Click(Sender: TObject);
20210:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
20211:     procedure EnlargeGutter1Click(Sender: TObject);
20212:     procedure Tetris1Click(Sender: TObject);
20213:     procedure ToDoList1Click(Sender: TObject);
20214:     procedure ProcessList1Click(Sender: TObject);
20215:     procedure MetricReport1Click(Sender: TObject);
20216:     procedure ProcessList1Click(Sender: TObject);
20217:     procedure TCPSockets1Click(Sender: TObject);
20218:     procedure ConfigUpdate1Click(Sender: TObject);
20219:     procedure ADOWorkbench1Click(Sender: TObject);
20220:     procedure SocketServer1Click(Sender: TObject);
20221:     procedure FormDemo1Click(Sender: TObject);
20222:     procedure Richedit1Click(Sender: TObject);
20223:     procedure SimpleBrowser1Click(Sender: TObject);
20224:     procedure DOSShell1Click(Sender: TObject);
20225:     procedure SynExport1Click(Sender: TObject);
20226:     procedure ExporttoRTF1Click(Sender: TObject);
20227:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
20228:     procedure SOAPTester1Click(Sender: TObject);
20229:     procedure Sniffer1Click(Sender: TObject);
20230:     procedure AutoDetectSyntax1Click(Sender: TObject);
20231:     procedure FPlot1Click(Sender: TObject);
20232:     procedure PasStyle1Click(Sender: TObject);
20233:     procedure Tutorial183RGBLED1Click(Sender: TObject);
20234:     procedure Reversi1Click(Sender: TObject);
20235:     procedure ManualmaXbox1Click(Sender: TObject);
20236:     procedure BlaisePascalMagazine1Click(Sender: TObject);
20237:     procedure AddToDo1Click(Sender: TObject);
20238:     procedure CreateGUID1Click(Sender: TObject);
20239:     procedure Tutorial27XML1Click(Sender: TObject);
20240:     procedure CreateDLLStub1Click(Sender: TObject);
20241:     procedure Tutorial28DLL1Click(Sender: TObject);');
20242:     procedure ResetKeyPressed;');
20243:     procedure KeyPressedFalse;
20244:     procedure FileChanges1Click(Sender: TObject);');
20245:     procedure OpenGLTry1Click(Sender: TObject);');
20246:     procedure AllUnitList1Click(Sender: TObject);');
20247:     procedure Tutorial29UMLClick(Sender: TObject);
20248:     procedure CreateHeader1Click(Sender: TObject);
20249:     procedure Oscilloscope1Click(Sender: TObject);');
20250:     procedure Tutorial30WOT1Click(Sender: TObject);');
20251:     procedure GetWebScript1Click(Sender: TObject);');
20252:     procedure Checkers1Click(Sender: TObject);');
20253:     procedure TaskMgr1Click(Sender: TObject);');
20254:     procedure WebCam1Click(Sender: TObject);');
20255:
20256:
20257: //-------------------------------------------------------------------------
20258: //*************mX4 Editor SynEdit Tools API ********************************
20259: //-------------------------------------------------------------------------
20260: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
```

```
20261: begin
20262:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
20263:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
20264:     Constructor Create( AOwner : TComponent)
20265:     SelStart', 'Integer', iptrw);
20266:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
20267:     Procedure UpdateCaret
20268:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
20269:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
20270:     Procedure BeginUndoBlock
20271:     Procedure BeginUpdate
20272:     Function CaretInView : Boolean
20273:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
20274:     Procedure Clear
20275:     Procedure ClearAll
20276:     Procedure ClearBookMark( BookMark : Integer)
20277:     Procedure ClearSelection
20278:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
20279:     Procedure ClearUndo
20280:     Procedure CopyToClipboard
20281:     Procedure CutToClipboard
20282:     Procedure DoCopyToClipboard( const SText : string)
20283:     Procedure EndUndoBlock
20284:     Procedure EndUpdate
20285:     Procedure EnsureCursorPosVisible
20286:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
20287:     Procedure FindMatchingBracket
20288:     Function GetMatchingBracket : TBufferCoord
20289:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
20290:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
20291:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
20292:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
20293:               : TSynHighlighterAttributes) : boolean
20294:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
20295:               var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
20296:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
20297:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
20298:     Procedure GotoBookMark( BookMark : Integer)
20299:     Procedure GotoLineAndCenter( ALine : Integer)
20300:     Function IdentChars : TSynIdentChars
20301:     Procedure InvalidateGutter
20302:     Procedure InvalidateGutterLine( aLine : integer)
20303:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
20304:     Procedure InvalidateLine( Line : integer)
20305:     Procedure InvalidateLines( FirstLine, LastLine : integer)
20306:     Procedure InvalidateSelection
20307:     Function IsBookmark( BookMark : integer) : boolean
20308:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
20309:     Procedure LockUndo
20310:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
20311:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
20312:     Function LineToRow( aLine : integer) : integer
20313:     Function RowToLine( aRow : integer) : integer
20314:     Function NextWordPos : TBufferCoord
20315:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
20316:     Procedure PasteFromClipboard
20317:     Function WordStart : TBufferCoord
20318:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
20319:     Function WordEnd : TBufferCoord
20320:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
20321:     Function PrevWordPos : TBufferCoord
20322:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
20323:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
20324:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
20325:     Procedure Redo
20326:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
20327:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
20328:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
20329:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
20330:     Procedure SelectAll
20331:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
20332:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
20333:     Procedure SetDefaultKeystrokes
20334:     Procedure SetSelWord
20335:     Procedure SetWordBlock( Value : TBufferCoord)
20336:     Procedure Undo
20337:     Procedure UnlockUndo
20338:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
20339:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
20340:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
20341:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
20342:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
20343:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
20344:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
20345:     Procedure AddFocusControl( aControl : TWinControl)
20346:     Procedure RemoveFocusControl( aControl : TWinControl)
20347:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
20348:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
20349:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
```

```
20350:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
20351:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
20352:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
20353:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
20354:     Procedure RemoveLinesPointer
20355:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
20356:     Procedure UnHookTextBuffer
20357:     BlockBegin', 'TBufferCoord', iptrw);
20358:     BlockEnd', 'TBufferCoord', iptrw);
20359:     CanPaste', 'Boolean', iptr);
20360:     CanRedo', 'boolean', iptr);
20361:     CanUndo', 'boolean', iptr);
20362:     CaretX', 'Integer', iptrw);
20363:     CaretY', 'Integer', iptrw);
20364:     CaretXY', 'TBufferCoord', iptrw);
20365:     ActiveLineColor', 'TColor', iptrw);
20366:     DisplayX', 'Integer', iptr);
20367:     DisplayY', 'Integer', iptr);
20368:     DisplayXY', 'TDisplayCoord', iptr);
20369:     DisplayLineCount', 'integer', iptr);
20370:     CharsInWindow', 'Integer', iptr);
20371:     CharWidth', 'integer', iptr);
20372:     Font', 'TFont', iptrw);
20373:     GutterWidth', 'Integer', iptr);
20374:     Highlighter', 'TSynCustomHighlighter', iptrw);
20375:     LeftChar', 'Integer', iptrw);
20376:     LineHeight', 'integer', iptr);
20377:     LinesInWindow', 'Integer', iptr);
20378:     LineText', 'string', iptrw);
20379:     Lines', 'TStrings', iptrw);
20380:     Marks', 'TSynEditMarkList', iptr);
20381:     MaxScrollWidth', 'integer', iptrw);
20382:     Modified', 'Boolean', iptrw);
20383:     PaintLock', 'Integer', iptr);
20384:     ReadOnly', 'Boolean', iptrw);
20385:     SearchEngine', 'TSynEditSearchCustom', iptrw);
20386:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
20387:     SelTabBlock', 'Boolean', iptr);
20388:     SelTabLine', 'Boolean', iptr);
20389:     SelText', 'string', iptrw);
20390:     StateFlags', 'TSynStateFlags', iptr);
20391:     Text', 'string', iptrw);
20392:     TopLine', 'Integer', iptrw);
20393:     WordAtCursor', 'string', iptr);
20394:     WordAtMouse', 'string', iptr);
20395:     UndoList', 'TSynEditUndoList', iptr);
20396:     RedoList', 'TSynEditUndoList', iptr);
20397:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
20398:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
20399:     BorderStyle', 'TSynBorderStyle', iptrw);
20400:     ExtraLineSpacing', 'integer', iptrw);
20401:     Gutter', 'TSynGutter', iptrw);
20402:     HideSelection', 'boolean', iptrw);
20403:     InsertCaret', 'TSynEditCaretType', iptrw);
20404:     InsertMode', 'boolean', iptrw);
20405:     IsScrolling', 'Boolean', iptr);
20406:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
20407:     MaxUndo', 'Integer', iptrw);
20408:     Options', 'TSynEditorOptions', iptrw);
20409:     OverwriteCaret', 'TSynEditCaretType', iptrw);
20410:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
20411:     ScrollHintColor', 'TColor', iptrw);
20412:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
20413:     ScrollBars', 'TScrollStyle', iptrw);
20414:     SelectedColor', 'TSynSelectedColor', iptrw);
20415:     SelectionMode', 'TSynSelectionMode', iptrw);
20416:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
20417:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
20418:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
20419:     WordWrapGlyph', 'TSynGlyph', iptrw);
20420:     OnChange', 'TNotifyEvent', iptrw);
20421:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
20422:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
20423:     OnContextHelp', 'TContextHelpEvent', iptrw);
20424:     OnDropFiles', 'TDropFilesEvent', iptrw);
20425:     OnGutterClick', 'TGutterClickEvent', iptrw);
20426:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
20427:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
20428:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
20429:     OnPaint', 'TPaintEvent', iptrw);
20430:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
20431:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
20432:     OnReplaceText', 'TReplaceTextEvent', iptrw);
20433:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
20434:     OnStatusChange', 'TStatusChangeEvent', iptrw);
20435:     OnPaintTransient', 'TPaintTransient', iptrw);
20436:     OnScroll', 'TScrollEvent', iptrw);
20437:    end;
20438:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
```

```
20439:  Function GetPlaceableHighlighters : TSynHighlighterList
20440:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
20441:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
20442:  Procedure GetEditorCommandValues( Proc : TGetStrProc)
20443:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)
20444:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
20445:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
20446:  Function ConvertCodeStringToExtended( AString : String) : String
20447:  Function ConvertExtendedToCodeString( AString : String) : String
20448:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
20449:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
20450:  Function IndexToEditorCommand( const AIndex : Integer) : Integer
20451:
20452:   TSynEditorOption = (
20453:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
20454:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
20455:                                // preceding line
20456:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
20457:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
20458:                                //direction any more
20459:     eoDragDropEditing,         //Allows to select a block of text and drag it within document to another
20460:                                // location
20461:     eoDropFiles,               //Allows the editor accept OLE file drops
20462:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
20463:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
20464:     eoGroupUndo,               //When undoing/redoing actions, handle all continous changes the same kind
20465:                                // in one call
20466:                                //instead undoing/redoing each command separately
20467:     eoHalfPageScroll,          //When scrolling with page-up and page-down commands, only scroll a half
20468:                                //page at a time
20469:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
20470:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
20471:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
20472:     eoNoCaret,                 //Makes it so the caret is never visible
20473:     eoNoSelection,             //Disables selecting text
20474:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
20475:     eoScrollByOneLess,         //Forces scrolling to be one less
20476:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
20477:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
20478:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
20479:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
20480:     eoShowSpecialChars,        //Shows the special Characters
20481:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
20482:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
20483:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
20484:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
20485:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
20486:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
20487:
20488:     ********************************Important Editor Short Cuts*****************************);
20489:     Double click to select a word and count words with highlightning.
20490:     Triple click to select a line.
20491:     CTRL+SHIFT+click to extend a selection.
20492:     Drag with the ALT key down to select columns of text !!!
20493:     Drag and drop is supported.
20494:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
20495:     Type CTRL+A to select all.
20496:     Type CTRL+N to set a new line.
20497:     Type CTRL+T to delete a line or token. //Tokenizer
20498:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
20499:     Type CTRL+Shift+T to add ToDo in line and list.
20500:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
20501:     Type CTRL[0..9] to jump or get to bookmarks.
20502:     Type Home to position cursor at beginning of current line and End to position it at end of line.
20503:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
20504:     Page Up and Page Down work as expected.
20505:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
20506:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
20507:
20508: {$ Short Key Positions Ctrl<A-Z>: }
20509: def
20510:    <A> Select All
20511:    <B> Count Words
20512:    <C> Copy
20513:    <D> Internet Start
20514:    <E> Script List
20515:    <F> Find
20516:    <G> Goto
20517:    <H> Mark Line
20518:    <I> Interface List
20519:    <J> Code Completion
20520:    <K> Console
20521:    <L> Interface List Box
20522:    <M> Font Larger -
20523:    <N> New Line
20524:    <O> Open File
20525:    <P> Font Smaller +
20526:    <Q> Quit
20527:    <R> Replace
```

```
20528:    <S> Save!
20529:    <T> Delete Line
20530:    <U> Use Case Editor
20531:    <V> Paste
20532:    <W> URI Links
20533:    <X> Reserved for coding use internal
20534:    <Y> Delete Line
20535:    <Z> Undo
20536:
20537: ref
20538:    F1 Help
20539:    F2 Syntax Check
20540:    F3 Search Next
20541:    F4 New Instance
20542:    F5 Line Mark /Breakpoint
20543:    F6 Goto End
20544:    F7 Debug Step Into
20545:    F8 Debug Step Out
20546:    F9 Compile
20547:    F10 Menu
20548:    F11 Word Count Highlight
20549:    F12 Reserved for coding use internal
20550:
20551:  def ReservedWords: array[0..82] of string =
20552:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
20553:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
20554:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
20555:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
20556:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
20557:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
20558:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
20559:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
20560:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
20561:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
20562:      'public', 'published','def','ref','using','typedef','memo1','memo2','doc','maxform1';
20563:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
20564: //-----------------------------------------------------------------------------
20565: //*************End of mX4 Public  Tools API *********************************
20566: //-----------------------------------------------------------------------------
20567:
20568: Amount of Functions: 13140
20569: Amount of Procedures: 8079
20570: Amount of Constructors: 1301
20571: Totals of Calls: 22520
20572: SHA1: Win Exe 3.9.9.98 4C7A84045994D69DA0131D3E518D9B4901F20C5D
20573:
20574:
20575: ************************************************************
20576:  Doc Short Manual with 50 Tips!
20577: ************************************************************
20578: - Install: just save your maxboxdef.ini before and then extract the zip file!
20579: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
20580: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
20581: - Menu: With <Crtl><F3> you can search for code on examples
20582: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
20583: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
20584: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
20585:
20586: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
20587: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
20588: - Context Menu: You can printout your scripts as a pdf-file or html-export
20589: - Context: You do have a context menu with the right mouse click
20590:
20591: - Menu: With the UseCase Editor you can convert graphic formats too.
20592: - Menu: On menu Options you find Addons as compiled scripts
20593: - IDE: You don't need a mouse to handle maXbox, use shortcuts
20594: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
20595: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
20596: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
20597:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
20598:
20599: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
20600: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
20601: - Code: If you code a loop till key-pressed use function: isKeyPressed;
20602: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
20603: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
20604: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
20605:          to delete and Click and mark to drag a bookmark
20606: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
20607: - IDE: A file info with system and script information you find in menu Program/Information
20608: - IDE: After change the config file in help you can update changes in menu Help/Config Update
20609: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
20610: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
20611: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
20612: - Editor: Set Bookmarks to check your work in app or code
20613: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
20614: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
20615: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
20616:
```

```
20617: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
20618: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
20619: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
20620: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
20621: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
20622: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
20623: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
20624: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
20625:
20626: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
20627: - Add on when no browser is available start /Options/Add ons/Easy Browser
20628: - Add on SOAP Tester with SOP POST File
20629: - Add on IP Protocol Sniffer with List View
20630: - Add on OpenGL mX Robot Demo for android
20631:
20632: - Menu: Help/Tools as a Tool Section with DOS Opener
20633: - Menu Editor: export the code as RTF File
20634: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
20635: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
20636: - Context: Auto Detect of Syntax depending on file extension
20637: - Code: some Windows API function start with w in the name like wGetAtomName();
20638: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
20639: - IDE File Check with menu ..View/File Changes/...
20640: - Context: Create a Header with Create Header in Navigator List at right window
20641: - Code: use  SysErrorMessage to get a real Error Description, Ex.
20642:         RemoveDir('c:\NoSuchFolder') ;
20643:         writeln('System Error Message: '+ SysErrorMessage(GetLastError));
20644: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
20645:
20646: - using DLL example in maXbox: //function: {*****************************************}
20647:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
20648:                                  cb: DWORD): BOOL; //stdcall;;
20649:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
20650:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
20651:      External  'OpenProcess@kernel32.dll stdcall';
20652:
20653: GCC Compile Ex Script
20654: procedure TFormMain_btnCompileClick(Sender: TObject);
20655: begin
20656:   AProcess:= TProcess.Create(Nil);
20657:   try
20658:     AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
20659:       + ' -o "' + OpenDialog2.FileName + '"';
20660:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
20661:     AProcess.Execute;
20662:     Memo2.Lines.BeginUpdate;
20663:     Memo2.Lines.Clear;
20664:     Memo2.Lines.LoadFromStream(AProcess.Output);
20665:     Memo2.Lines.EndUpdate;
20666:   finally
20667:     AProcess.Free;
20668:   end;
20669: end;
20670:
20671:
20672: History Shell Hell
20673: PCT Precompile Technology , mX4 ScriptStudio
20674: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
20675: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
20676: emax layers: system-package-component-unit-class-function-block
20677: new keywords def ref using maXCalcF
20678: UML: use case act class state seq pac comp dep - lib lab
20679: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
20680: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
20681: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
20682: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
20683: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
20684: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
20685: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
20686: SendBuffer, Color+Caption hack, ComboSet,SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
20687: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
20688: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
20689: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
20690: Inno Install and Setup Routines
20691: Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
20692: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
20693: VfW (Video), FindFirst3, ResFiler, AssemblyCache
20694:
20695:
20696: https://unibe-ch.academia.edu/MaxKleiner
20697: www.slideshare.net/maxkleiner1
20698: http://www.scribd.com/max_kleiner
20699: http://www.delphiforfun.org/Programs/Utilities/index.htm
20700: http://www.slideshare.net/maxkleiner1
20701: http://s3.amazonaws.com/PreviewLinks/22959.html
20702: http://www.softwareschule.ch/arduino_training.pdf
20703: http://www.jrsoftware.org/isinfo.php
20704: http://www.be-precision.com/products/precision-builder/express/
20705: http://www.blaisepascal.eu/
```

```
20706:
20707:
20708: ************************************************************
20709:  unit List asm internal end
20710: ************************************************************
20711: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
20712: 02 unit SIRegister_IdStrings                   //Indy Sockets
20713: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
20714: 04 unit uPSI_fMain Functions;                   //maXbox Open Tools API
20715: 05 unit IFSI_WinForm1puzzle;                    //maXbox
20716: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
20717: 07 unit RegisterDateTimeLibrary_R(exec);       //Delphi
20718: 08 unit RIRegister_MathMax_Routines(exec);     //Jedi & Delphi
20719: 09 unit RIRegister_IdGlobal_Routines(exec);    //Indy Sockets
20720: 10 unit RIRegister_SysUtils_Routines(Exec);    //Delphi
20721: 11 unit uPSI_IdTCPConnection;                  //Indy some functions
20722: 12 unit uPSCompiler.pas;                       //PS kernel functions
20723: 13 unit uPSI_DBCommon;                         //DB Common_Routines and Types
20724: 14 unit uPSI_Printers.pas                      //Delphi VCL
20725: 15 unit uPSI_MPlayer.pas                      //Delphi VCL
20726: 16 unit uPSC_comobj;                          //COM Functions
20727: 17 unit uPSI_Clipbrd;                         //Delphi VCL
20728: 18 unit Filectrl in IFSI_SysUtils_max;        //VCL Runtime
20729: 19 unit uPSI_SqlExpr;                        //DBX3
20730: 20 unit uPSI_ADODB;                          //ADODB
20731: 21 unit uPSI_StrHlpr;                        //String Helper Routines
20732: 22 unit uPSI_DateUtils;                      //Expansion to DateTimeLib
20733: 23 unit uPSI_FileUtils;                      //Expansion to Sys/File Utils
20734: 24 unit JUtils / gsUtils;                    //Jedi / Metabase
20735: 25 unit JvFunctions_max;                     //Jedi Functions
20736: 26 unit HTTPParser;                          //Delphi VCL
20737: 27 unit HTTPUtil;                           //Delphi VCL
20738: 28 unit uPSI_XMLUtil;                       //Delphi VCL
20739: 29 unit uPSI_SOAPHTTPClient;                 //Delphi VCL SOAP WebService V3.5
20740: 30 unit uPSI_Contnrs;                       //Delphi RTL Container of Classes
20741: 31 unit uPSI_MaskUtils;                     //RTL Edit and Mask functions
20742: 32 unit uPSI_MyBigInt;                      //big integer class with Math
20743: 33 unit uPSI_ConvUtils;                     //Delphi VCL Conversions engine
20744: 34 unit Types_Variants;                     //Delphi\Win32\rtl\sys
20745: 35 unit uPSI_IdHashSHA1;                    //Indy Crypto Lib
20746: 36 unit uPSI_IdHashMessageDigest           //Indy Crypto;
20747: 37 unit uPSI_IdASN1Util;                    //Indy ASN1Utility Routines;
20748: 38 unit uPSI_IdLogFile;                     //Indy Logger from LogBase
20749: 39 unit uPSI_IdIcmpClient;                  //Indy Ping ICMP
20750: 40 unit uPSI_IdHashMessageDigest_max       //Indy Crypto &OpenSSL;
20751: 41 unit uPSI_FileCtrl;                      //Delphi RTL
20752: 42 unit uPSI_Outline;                       //Delphi VCL
20753: 43 unit uPSI_ScktComp;                      //Delphi RTL
20754: 44 unit uPSI_Calendar;                      //Delphi VCL
20755: 45 unit uPSI_VListView                      //VListView;
20756: 46 unit uPSI_DBGrids;                       //Delphi VCL
20757: 47 unit uPSI_DBCtrls;                       //Delphi VCL
20758: 48 unit ide_debugoutput;                    //maXbox
20759: 49 unit uPSI_ComCtrls;                      //Delphi VCL
20760: 50 unit uPSC_stdctrls+;                     //Delphi VCL
20761: 51 unit uPSI_Dialogs;                       //Delphi VCL
20762: 52 unit uPSI_StdConvs;                      //Delphi RTL
20763: 53 unit uPSI_DBClient;                      //Delphi RTL
20764: 54 unit uPSI_DBPlatform;                    //Delphi RTL
20765: 55 unit uPSI_Provider;                      //Delphi RTL
20766: 56 unit uPSI_FMTBcd;                        //Delphi RTL
20767: 57 unit uPSI_DBCGrids;                      //Delphi VCL
20768: 58 unit uPSI_CDSUtil;                       //MIDAS
20769: 59 unit uPSI_VarHlpr;                       //Delphi RTL
20770: 60 unit uPSI_ExtDlgs;                       //Delphi VCL
20771: 61 unit sdpStopwatch;                       //maXbox
20772: 62 unit uPSI_JclStatistics;                 //JCL
20773: 63 unit uPSI_JclLogic;                      //JCL
20774: 64 unit uPSI_JclMiscel;                     //JCL
20775: 65 unit uPSI_JclMath_max;                   //JCL RTL
20776: 66 unit uPSI_uTPLb_StreamUtils;             //LockBox 3
20777: 67 unit uPSI_MathUtils;                     //BCB
20778: 68 unit uPSI_JclMultimedia;                 //JCL
20779: 69 unit uPSI_WideStrUtils;                  //Delphi API/RTL
20780: 70 unit uPSI_GraphUtil;                     //Delphi RTL
20781: 71 unit uPSI_TypeTrans;                     //Delphi RTL
20782: 72 unit uPSI_HTTPApp;                       //Delphi VCL
20783: 73 unit uPSI_DBWeb;                         //Delphi VCL
20784: 74 unit uPSI_DBBdeWeb;                      //Delphi VCL
20785: 75 unit uPSI_DBXpressWeb;                   //Delphi VCL
20786: 76 unit uPSI_ShadowWnd;                     //Delphi VCL
20787: 77 unit uPSI_ToolWin;                       //Delphi VCL
20788: 78 unit uPSI_Tabs;                          //Delphi VCL
20789: 79 unit uPSI_JclGraphUtils;                 //JCL
20790: 80 unit uPSI_JclCounter;                    //JCL
20791: 81 unit uPSI_JclSysInfo;                    //JCL
20792: 82 unit uPSI_JclSecurity;                   //JCL
20793: 83 unit uPSI_JclFileUtils;                  //JCL
20794: 84 unit uPSI_IdUserAccounts;                //Indy
```

```
20795: 85 unit uPSI_IdAuthentication;                //Indy
20796: 86 unit uPSI_uTPLb_AES;                       //LockBox 3
20797: 87 unit uPSI_IdHashSHA1;                      //LockBox 3
20798: 88 unit uTPLb_BlockCipher;                    //LockBox 3
20799: 89 unit uPSI_ValEdit.pas;                     //Delphi VCL
20800: 90 unit uPSI_JvVCLUtils;                      //JCL
20801: 91 unit uPSI_JvDBUtil;                        //JCL
20802: 92 unit uPSI_JvDBUtils;                       //JCL
20803: 93 unit uPSI_JvAppUtils;                      //JCL
20804: 94 unit uPSI_JvCtrlUtils;                     //JCL
20805: 95 unit uPSI_JvFormToHtml;                    //JCL
20806: 96 unit uPSI_JvParsing;                       //JCL
20807: 97 unit uPSI_SerDlgs;                        //Toolbox
20808: 98 unit uPSI_Serial;                         //Toolbox
20809: 99 unit uPSI_JvComponent;                    //JCL
20810: 100 unit uPSI_JvCalc;                        //JCL
20811: 101 unit uPSI_JvBdeUtils;                    //JCL
20812: 102 unit uPSI_JvDateUtil;                    //JCL
20813: 103 unit uPSI_JvGenetic;                     //JCL
20814: 104 unit uPSI_JclBase;                       //JCL
20815: 105 unit uPSI_JvUtils;                       //JCL
20816: 106 unit uPSI_JvStrUtil;                     //JCL
20817: 107 unit uPSI_JvStrUtils;                    //JCL
20818: 108 unit uPSI_JvFileUtil;                    //JCL
20819: 109 unit uPSI_JvMemoryInfos;                 //JCL
20820: 110 unit uPSI_JvComputerInfo;                //JCL
20821: 111 unit uPSI_JvgCommClasses;                //JCL
20822: 112 unit uPSI_JvgLogics;                     //JCL
20823: 113 unit uPSI_JvLED;                         //JCL
20824: 114 unit uPSI_JvTurtle;                      //JCL
20825: 115 unit uPSI_SortThds; unit uPSI_ThSort;    //maXbox
20826: 116 unit uPSI_JvgUtils;                      //JCL
20827: 117 unit uPSI_JvExprParser;                  //JCL
20828: 118 unit uPSI_HexDump;                       //Borland
20829: 119 unit uPSI_DBLogDlg;                      //VCL
20830: 120 unit uPSI_SqlTimSt;                      //RTL
20831: 121 unit uPSI_JvHtmlParser;                  //JCL
20832: 122 unit uPSI_JvgXMLSerializer;              //JCL
20833: 123 unit uPSI_JvJCLUtils;                    //JCL
20834: 124 unit uPSI_JvStrings;                     //JCL
20835: 125 unit uPSI_uTPLb_IntegerUtils;            //TurboPower
20836: 126 unit uPSI_uTPLb_HugeCardinal;            //TurboPower
20837: 127 unit uPSI_uTPLb_HugeCardinalUtils;       //TurboPower
20838: 128 unit uPSI_SynRegExpr;                    //SynEdit
20839: 129 unit uPSI_StUtils;                       //SysTools4
20840: 130 unit uPSI_StToHTML;                      //SysTools4
20841: 131 unit uPSI_StStrms;                       //SysTools4
20842: 132 unit uPSI_StFIN;                         //SysTools4
20843: 133 unit uPSI_StAstroP;                      //SysTools4
20844: 134 unit uPSI_StStat;                        //SysTools4
20845: 135 unit uPSI_StNetCon;                      //SysTools4
20846: 136 unit uPSI_StDecMth;                      //SysTools4
20847: 137 unit uPSI_StOStr;                        //SysTools4
20848: 138 unit uPSI_StPtrns;                       //SysTools4
20849: 139 unit uPSI_StNetMsg;                      //SysTools4
20850: 140 unit uPSI_StMath;                        //SysTools4
20851: 141 unit uPSI_StExpEng;                      //SysTools4
20852: 142 unit uPSI_StCRC;                         //SysTools4
20853: 143 unit uPSI_StExport,                      //SysTools4
20854: 144 unit uPSI_StExpLog,                      //SysTools4
20855: 145 unit uPSI_ActnList;                      //Delphi VCL
20856: 146 unit uPSI_jpeg;                          //Borland
20857: 147 unit uPSI_StRandom;                      //SysTools4
20858: 148 unit uPSI_StDict;                        //SysTools4
20859: 149 unit uPSI_StBCD;                         //SysTools4
20860: 150 unit uPSI_StTxtDat;                      //SysTools4
20861: 151 unit uPSI_StRegEx;                       //SysTools4
20862: 152 unit uPSI_IMouse;                        //VCL
20863: 153 unit uPSI_SyncObjs;                      //VCL
20864: 154 unit uPSI_AsyncCalls;                    //Hausladen
20865: 155 unit uPSI_ParallelJobs;                  //Saraiva
20866: 156 unit uPSI_Variants;                      //VCL
20867: 157 unit uPSI_VarCmplx;                      //VCL Wolfram
20868: 158 unit uPSI_DTDSchema;                     //VCL
20869: 159 unit uPSI_ShLwApi;                       //Brakel
20870: 160 unit uPSI_IBUtils;                       //VCL
20871: 161 unit uPSI_CheckLst;                      //VCL
20872: 162 unit uPSI_JvSimpleXml;                   //JCL
20873: 163 unit uPSI_JclSimpleXml;                  //JCL
20874: 164 unit uPSI_JvXmlDatabase;                 //JCL
20875: 165 unit uPSI_JvMaxPixel;                    //JCL
20876: 166 unit uPSI_JvItemsSearchs;                //JCL
20877: 167 unit uPSI_StExpEng2;                     //SysTools4
20878: 168 unit uPSI_StGenLog;                      //SysTools4
20879: 169 unit uPSI_JvLogFile;                     //Jcl
20880: 170 unit uPSI_CPort;                         //ComPort Lib v4.11
20881: 171 unit uPSI_CPortCtl;                      //ComPort
20882: 172 unit uPSI_CPortEsc;                      //ComPort
20883: 173 unit BarCodeScaner;                      //ComPort
```

```
20884: 174 unit uPSI_JvGraph;                        //JCL
20885: 175 unit uPSI_JvComCtrls;                    //JCL
20886: 176 unit uPSI_GUITesting;                    //D Unit
20887: 177 unit uPSI_JvFindFiles;                   //JCL
20888: 178 unit uPSI_StSystem;                      //SysTools4
20889: 179 unit uPSI_JvKeyboardStates;             //JCL
20890: 180 unit uPSI_JvMail;                       //JCL
20891: 181 unit uPSI_JclConsole;                   //JCL
20892: 182 unit uPSI_JclLANMan;                    //JCL
20893: 183 unit uPSI_IdCustomHTTPServer;           //Indy
20894: 184 unit IdHTTPServer                       //Indy
20895: 185 unit uPSI_IdTCPServer;                  //Indy
20896: 186 unit uPSI_IdSocketHandle;               //Indy
20897: 187 unit uPSI_IdIOHandlerSocket;            //Indy
20898: 188 unit IdIOHandler;                       //Indy
20899: 189 unit uPSI_cutils;                       //Bloodshed
20900: 190 unit uPSI_BoldUtils;                    //boldsoft
20901: 191 unit uPSI_IdSimpleServer;               //Indy
20902: 192 unit uPSI_IdSSLOpenSSL;                 //Indy
20903: 193 unit uPSI_IdMultipartFormData;          //Indy
20904: 194 unit uPSI_SynURIOpener;                 //SynEdit
20905: 195 unit uPSI_PerlRegEx;                    //PCRE
20906: 196 unit uPSI_IdHeaderList;                 //Indy
20907: 197 unit uPSI_StFirst;                      //SysTools4
20908: 198 unit uPSI_JvCtrls;                      //JCL
20909: 199 unit uPSI_IdTrivialFTPBase;             //Indy
20910: 200 unit uPSI_IdTrivialFTP;                 //Indy
20911: 201 unit uPSI_IdUDPBase;                    //Indy
20912: 202 unit uPSI_IdUDPClient;                  //Indy
20913: 203 unit uPSI_utypes;                       //for DMath.DLL
20914: 204 unit uPSI_ShellAPI;                     //Borland
20915: 205 unit uPSI_IdRemoteCMDClient;            //Indy
20916: 206 unit uPSI_IdRemoteCMDServer;            //Indy
20917: 207 unit IdRexecServer;                     //Indy
20918: 208 unit IdRexec; (unit uPSI_IdRexec;)      //Indy
20919: 209 unit IdUDPServer;                       //Indy
20920: 210 unit IdTimeUDPServer;                   //Indy
20921: 211 unit IdTimeServer;                      //Indy
20922: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
20923: 213 unit uPSI_IdIPWatch;                    //Indy
20924: 214 unit uPSI_IdIrcServer;                  //Indy
20925: 215 unit uPSI_IdMessageCollection;          //Indy
20926: 216 unit uPSI_cPEM;                         //Fundamentals 4
20927: 217 unit uPSI_cFundamentUtils;              //Fundamentals 4
20928: 218 unit uPSI_uwinplot;                     //DMath
20929: 219 unit uPSI_xrtl_util_CPUUtils;           //ExtentedRTL
20930: 220 unit uPSI_GR32_System;                  //Graphics32
20931: 221 unit uPSI_cFileUtils;                   //Fundamentals 4
20932: 222 unit uPSI_cDateTime; (timemachine)      //Fundamentals 4
20933: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
20934: 224 unit uPSI_cRandom;                      //Fundamentals 4
20935: 225 unit uPSI_ueval;                        //DMath
20936: 226 unit uPSI_xrtl_net_URIUtils;            //ExtendedRTL
20937: 227 unit xrtl_net_URIUtils;                 //ExtendedRTL
20938: 228 unit uPSI_ufft;  (FFT)                  //DMath
20939: 229 unit uPSI_DBXChannel;                   //Delphi
20940: 230 unit uPSI_DBXIndyChannel;               //Delphi Indy
20941: 231 unit uPSI_xrtl_util_COMCat;             //ExtendedRTL
20942: 232 unit uPSI_xrtl_util_StrUtils;           //ExtendedRTL
20943: 233 unit uPSI_xrtl_util_VariantUtils;       //ExtendedRTL
20944: 234 unit uPSI_xrtl_util_FileUtils;          //ExtendedRTL
20945: 235 unit xrtl_util_Compat;                  //ExtendedRTL
20946: 236 unit uPSI_OleAuto;                      //Borland
20947: 237 unit uPSI_xrtl_util_COMUtils;           //ExtendedRTL
20948: 238 unit uPSI_CmAdmCtl;                     //Borland
20949: 239 unit uPSI_ValEdit2;                     //VCL
20950: 240 unit uPSI_GR32;  //Graphics32          //Graphics32
20951: 241 unit uPSI_GR32_Image;                   //Graphics32
20952: 242 unit uPSI_xrtl_util_TimeUtils;          //ExtendedRTL
20953: 243 unit uPSI_xrtl_util_TimeZone;           //ExtendedRTL
20954: 244 unit uPSI_xrtl_util_TimeStamp;          //ExtendedRTL
20955: 245 unit uPSI_xrtl_util_Map;                //ExtendedRTL
20956: 246 unit uPSI_xrtl_util_Set;                //ExtendedRTL
20957: 247 unit uPSI_CPortMonitor;                 //ComPort
20958: 248 unit uPSI_StIniStm;                     //SysTools4
20959: 249 unit uPSI_GR32_ExtImage;                //Graphics32
20960: 250 unit uPSI_GR32_OrdinalMaps;             //Graphics32
20961: 251 unit uPSI_GR32_Rasterizers;             //Graphics32
20962: 252 unit uPSI_xrtl_util_Exception;          //ExtendedRTL
20963: 253 unit uPSI_xrtl_util_Value;              //ExtendedRTL
20964: 254 unit uPSI_xrtl_util_Compare;            //ExtendedRTL
20965: 255 unit uPSI_FlatSB;                       //VCL
20966: 256 unit uPSI_JvAnalogClock;                //JCL
20967: 257 unit uPSI_JvAlarms;                     //JCL
20968: 258 unit uPSI_JvSQLS;                       //JCL
20969: 259 unit uPSI_JvDBSecur;                    //JCL
20970: 260 unit uPSI_JvDBQBE;                      //JCL
20971: 261 unit uPSI_JvStarfield;                  //JCL
20972: 262 unit uPSI_JVCLMiscal;                   //JCL
```

```
20973: 263 unit uPSI_JvProfiler32;                  //JCL
20974: 264 unit uPSI_JvDirectories,                 //JCL
20975: 265 unit uPSI_JclSchedule,                   //JCL
20976: 266 unit uPSI_JclSvcCtrl,                    //JCL
20977: 267 unit uPSI_JvSoundControl,               //JCL
20978: 268 unit uPSI_JvBDESQLScript,               //JCL
20979: 269 unit uPSI_JvgDigits,                    //JCL>
20980: 270 unit uPSI_ImgList;                      //TCustomImageList
20981: 271 unit uPSI_JclMIDI;                      //JCL>
20982: 272 unit uPSI_JclWinMidi;                   //JCL>
20983: 273 unit uPSI_JclNTFS;                      //JCL>
20984: 274 unit uPSI_JclAppInst;                   //JCL>
20985: 275 unit uPSI_JvRle;                        //JCL>
20986: 276 unit uPSI_JvRas32;                      //JCL>
20987: 277 unit uPSI_JvImageDrawThread,            //JCL>
20988: 278 unit uPSI_JvImageWindow,                //JCL>
20989: 279 unit uPSI_JvTransparentForm;            //JCL>
20990: 280 unit uPSI_JvWinDialogs;                 //JCL>
20991: 281 unit uPSI_JvSimLogic,                   //JCL>
20992: 282 unit uPSI_JvSimIndicator,               //JCL>
20993: 283 unit uPSI_JvSimPID,                     //JCL>
20994: 284 unit uPSI_JvSimPIDLinker,               //JCL>
20995: 285 unit uPSI_IdRFCReply;                   //Indy
20996: 286 unit uPSI_IdIdent;                      //Indy
20997: 287 unit uPSI_IdIdentServer;                //Indy
20998: 288 unit uPSI_JvPatchFile;                  //JCL
20999: 289 unit uPSI_StNetPfm;                     //SysTools4
21000: 290 unit uPSI_StNet;                        //SysTools4
21001: 291 unit uPSI_JclPeImage;                   //JCL
21002: 292 unit uPSI_JclPrint;                     //JCL
21003: 293 unit uPSI_JclMime;                      //JCL
21004: 294 unit uPSI_JvRichEdit;                   //JCL
21005: 295 unit uPSI_JvDBRichEd;                   //JCL
21006: 296 unit uPSI_JvDice;                       //JCL
21007: 297 unit uPSI_JvFloatEdit;                  //JCL 3.9.8
21008: 298 unit uPSI_JvDirFrm;                     //JCL
21009: 299 unit uPSI_JvDualList;                   //JCL
21010: 300 unit uPSI_JvSwitch;                     ////JCL
21011: 301 unit uPSI_JvTimerLst;                   ////JCL
21012: 302 unit uPSI_JvMemTable;                   //JCL
21013: 303 unit uPSI_JvObjStr;                     //JCL
21014: 304 unit uPSI_StLArr;                       //SysTools4
21015: 305 unit uPSI_StWmDCpy;                     //SysTools4
21016: 306 unit uPSI_StText;                       //SysTools4
21017: 307 unit uPSI_StNTLog;                      //SysTools4
21018: 308 unit uPSI_xrtl_math_Integer;            //ExtendedRTL
21019: 309 unit uPSI_JvImagPrvw;                   //JCL
21020: 310 unit uPSI_JvFormPatch;                  //JCL
21021: 311 unit uPSI_JvPicClip;                    //JCL
21022: 312 unit uPSI_JvDataConv;                   //JCL
21023: 313 unit uPSI_JvCpuUsage;                   //JCL
21024: 314 unit uPSI_JclUnitConv_mX2;              //JCL
21025: 315 unit JvDualListForm;                    //JCL
21026: 316 unit uPSI_JvCpuUsage2;                  //JCL
21027: 317 unit uPSI_JvParserForm;                 //JCL
21028: 318 unit uPSI_JvJanTreeView;                //JCL
21029: 319 unit uPSI_JvTransLED;                   //JCL
21030: 320 unit uPSI_JvPlaylist;                   //JCL
21031: 321 unit uPSI_JvFormAutoSize;               //JCL
21032: 322 unit uPSI_JvYearGridEditForm;           //JCL
21033: 323 unit uPSI_JvMarkupCommon;               //JCL
21034: 324 unit uPSI_JvChart;                      //JCL
21035: 325 unit uPSI_JvXPCore;                     //JCL
21036: 326 unit uPSI_JvXPCoreUtils;                //JCL
21037: 327 unit uPSI_StatsClasses;                 //mX4
21038: 328 unit uPSI_ExtCtrls2;                    //VCL
21039: 329 unit uPSI_JvUrlGrabbers;                //JCL
21040: 330 unit uPSI_JvXmlTree;                    //JCL
21041: 331 unit uPSI_JvWavePlayer;                 //JCL
21042: 332 unit uPSI_JvUnicodeCanvas;              //JCL
21043: 333 unit uPSI_JvTFUtils;                    //JCL
21044: 334 unit uPSI_IdServerIOHandler;            //Indy
21045: 335 unit uPSI_IdServerIOHandlerSocket;      //Indy
21046: 336 unit uPSI_IdMessageCoder;               //Indy
21047: 337 unit uPSI_IdMessageCoderMIME;           //Indy
21048: 338 unit uPSI_IdMIMETypes;                  //Indy
21049: 339 unit uPSI_JvConverter;                  //JCL
21050: 340 unit uPSI_JvCsvParse;                   //JCL
21051: 341 unit uPSI_umath;  unit uPSI_ugamma;     //DMath
21052: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
21053: 343 unit uPSI_JvDBGridExport;               //JCL
21054: 344 unit uPSI_JvgExport;                    //JCL
21055: 345 unit uPSI_JvSerialMaker;                //JCL
21056: 346 unit uPSI_JvWin32;                      //JCL
21057: 347 unit uPSI_JvPaintFX;                    //JCL
21058: 348 unit uPSI_JvOracleDataSet; (beta)       //JCL
21059: 349 unit uPSI_JvValidators; (preview)       //JCL
21060: 350 unit uPSI_JvNTEventLog;                 //JCL
21061: 351 unit uPSI_ShellZipTool;                 //mX4
```

```
21062: 352 unit uPSI_JvJoystick;                    //JCL
21063: 353 unit uPSI_JvMailSlots;                   //JCL
21064: 354 unit uPSI_JclComplex;                    //JCL
21065: 355 unit uPSI_SynPdf;                        //Synopse
21066: 356 unit uPSI_Registry;                      //VCL
21067: 357 unit uPSI_TlHelp32;                      //VCL
21068: 358 unit uPSI_JclRegistry;                   //JCL
21069: 359 unit uPSI_JvAirBrush;                    //JCL
21070: 360 unit uPSI_mORMotReport;                  //Synopse
21071: 361 unit uPSI_JclLocales;                    //JCL
21072: 362 unit uPSI_SynEdit;                       //SynEdit
21073: 363 unit uPSI_SynEditTypes;                  //SynEdit
21074: 364 unit uPSI_SynMacroRecorder;              //SynEdit
21075: 365 unit uPSI_LongIntList;                   //SynEdit
21076: 366 unit uPSI_devcutils;                     //DevC
21077: 367 unit uPSI_SynEditMiscClasses;            //SynEdit
21078: 368 unit uPSI_SynEditRegexSearch;            //SynEdit
21079: 369 unit uPSI_SynEditHighlighter;            //SynEdit
21080: 370 unit uPSI_SynHighlighterPas;             //SynEdit
21081: 371 unit uPSI_JvSearchFiles;                 //JCL
21082: 372 unit uPSI_SynHighlighterAny;             //Lazarus
21083: 373 unit uPSI_SynEditKeyCmds;                //SynEdit
21084: 374 unit uPSI_SynEditMiscProcs,              //SynEdit
21085: 375 unit uPSI_SynEditKbdHandler              //SynEdit
21086: 376 unit uPSI_JvAppInst,                     //JCL
21087: 377 unit uPSI_JvAppEvent;                    //JCL
21088: 378 unit uPSI_JvAppCommand;                  //JCL
21089: 379 unit uPSI_JvAnimTitle;                   //JCL
21090: 380 unit uPSI_JvAnimatedImage;               //JCL
21091: 381 unit uPSI_SynEditExport;                 //SynEdit
21092: 382 unit uPSI_SynExportHTML;                 //SynEdit
21093: 383 unit uPSI_SynExportRTF;                  //SynEdit
21094: 384 unit uPSI_SynEditSearch;                 //SynEdit
21095: 385 unit uPSI_fMain_back                     //maXbox;
21096: 386 unit uPSI_JvZoom;                        //JCL
21097: 387 unit uPSI_PMrand;                        //PM
21098: 388 unit uPSI_JvSticker;                     //JCL
21099: 389 unit uPSI_XmlVerySimple;                 //mX4
21100: 390 unit uPSI_Services;                      //ExtPascal
21101: 391 unit uPSI_ExtPascalUtils;                //ExtPascal
21102: 392 unit uPSI_SocketsDelphi;                 //ExtPascal
21103: 393 unit uPSI_StBarC;                        //SysTools
21104: 394 unit uPSI_StDbBarC;                      //SysTools
21105: 395 unit uPSI_StBarPN;                       //SysTools
21106: 396 unit uPSI_StDbPNBC;                      //SysTools
21107: 397 unit uPSI_StDb2DBC;                      //SysTools
21108: 398 unit uPSI_StMoney;                       //SysTools
21109: 399 unit uPSI_JvForth;                       //JCL
21110: 400 unit uPSI_RestRequest;                   //mX4
21111: 401 unit uPSI_HttpRESTConnectionIndy;        //mX4
21112: 402 unit uPSI_JvXmlDatabase; //update       //JCL
21113: 403 unit uPSI_StAstro;                       //SysTools
21114: 404 unit uPSI_StSort;                        //SysTools
21115: 405 unit uPSI_StDate;                        //SysTools
21116: 406 unit uPSI_StDateSt;                      //SysTools
21117: 407 unit uPSI_StBase;                        //SysTools
21118: 408 unit uPSI_StVInfo;                       //SysTools
21119: 409 unit uPSI_JvBrowseFolder;                //JCL
21120: 410 unit uPSI_JvBoxProcs;                    //JCL
21121: 411 unit uPSI_urandom; (unit uranuvag;)      //DMath
21122: 412 unit uPSI_usimann; (unit ugenalg;)       //DMath
21123: 413 unit uPSI_JvHighlighter;                 //JCL
21124: 414 unit uPSI_Diff;                          //mX4
21125: 415 unit uPSI_SpringWinAPI;                  //DSpring
21126: 416 unit uPSI_StBits;                        //SysTools
21127: 417 unit uPSI_TomDBQue;                      //mX4
21128: 418 unit uPSI_MultilangTranslator;           //mX4
21129: 419 unit uPSI_HyperLabel;                    //mX4
21130: 420 unit uPSI_Starter;                       //mX4
21131: 421 unit uPSI_FileAssocs;                    //devC
21132: 422 unit uPSI_devFileMonitorX;               //devC
21133: 423 unit uPSI_devrun;                        //devC
21134: 424 unit uPSI_devExec;                       //devC
21135: 425 unit uPSI_oysUtils;                      //devC
21136: 426 unit uPSI_DosCommand;                    //devC
21137: 427 unit uPSI_CppTokenizer;                  //devC
21138: 428 unit uPSI_JvHLParser;                    //devC
21139: 429 unit uPSI_JclMapi;                       //JCL
21140: 430 unit uPSI_JclShell;                      //JCL
21141: 431 unit uPSI_JclCOM;                        //JCL
21142: 432 unit uPSI_GR32_Math;                     //Graphics32
21143: 433 unit uPSI_GR32_LowLevel;                 //Graphics32
21144: 434 unit uPSI_SimpleHl;                      //mX4
21145: 435 unit uPSI_GR32_Filters,                  //Graphics32
21146: 436 unit uPSI_GR32_VectorMaps;               //Graphics32
21147: 437 unit uPSI_cXMLFunctions;                 //Fundamentals 4
21148: 438 unit uPSI_JvTimer;                       //JCL
21149: 439 unit uPSI_cHTTPUtils;                    //Fundamentals 4
21150: 440 unit uPSI_cTLSUtils;                     //Fundamentals 4
```

```
21151: 441 unit uPSI_JclGraphics;                    //JCL
21152: 442 unit uPSI_JclSynch;                       //JCL
21153: 443 unit uPSI_IdTelnet;                       //Indy
21154: 444 unit uPSI_IdTelnetServer,                 //Indy
21155: 445 unit uPSI_IdEcho,                         //Indy
21156: 446 unit uPSI_IdEchoServer,                   //Indy
21157: 447 unit uPSI_IdEchoUDP,                      //Indy
21158: 448 unit uPSI_IdEchoUDPServer,                //Indy
21159: 449 unit uPSI_IdSocks,                       //Indy
21160: 450 unit uPSI_IdAntiFreezeBase;              //Indy
21161: 451 unit uPSI_IdHostnameServer;              //Indy
21162: 452 unit uPSI_IdTunnelCommon,                //Indy
21163: 453 unit uPSI_IdTunnelMaster,                //Indy
21164: 454 unit uPSI_IdTunnelSlave,                 //Indy
21165: 455 unit uPSI_IdRSH,                         //Indy
21166: 456 unit uPSI_IdRSHServer,                   //Indy
21167: 457 unit uPSI_Spring_Cryptography_Utils;     //Spring4Delphi
21168: 458 unit uPSI_MapReader,                     //devC
21169: 459 unit uPSI_LibTar,                        //devC
21170: 460 unit uPSI_IdStack;                       //Indy
21171: 461 unit uPSI_IdBlockCipherIntercept;        //Indy
21172: 462 unit uPSI_IdChargenServer;               //Indy
21173: 463 unit uPSI_IdFTPServer,                   //Indy
21174: 464 unit uPSI_IdException,                   //Indy
21175: 465 unit uPSI_utexplot;                      //DMath
21176: 466 unit uPSI_uwinstr;                       //DMath
21177: 467 unit uPSI_VarRecUtils;                   //devC
21178: 468 unit uPSI_JvStringListToHtml,            //JCL
21179: 469 unit uPSI_JvStringHolder,                //JCL
21180: 470 unit uPSI_IdCoder;                       //Indy
21181: 471 unit uPSI_SynHighlighterDfm;             //Synedit
21182: 472 unit uHighlighterProcs; in 471           //Synedit
21183: 473 unit uPSI_LazFileUtils,                  //LCL
21184: 474 unit uPSI_IDECmdLine;                    //LCL
21185: 475 unit uPSI_lazMasks;                      //LCL
21186: 476 unit uPSI_ip_misc;                       //mX4
21187: 477 unit uPSI_Barcode;                       //LCL
21188: 478 unit uPSI_SimpleXML;                     //LCL
21189: 479 unit uPSI_JclIniFiles;                   //JCL
21190: 480 unit uPSI_D2XXUnit;  {$X-}               //FTDI
21191: 481 unit uPSI_JclDateTime;                   //JCL
21192: 482 unit uPSI_JclEDI;                        //JCL
21193: 483 unit uPSI_JclMiscel2;                    //JCL
21194: 484 unit uPSI_JclValidation;                 //JCL
21195: 485 unit uPSI_JclAnsiStrings; {-PString}     //JCL
21196: 486 unit uPSI_SynEditMiscProcs2;             //Synedit
21197: 487 unit uPSI_JclStreams;                    //JCL
21198: 488 unit uPSI_QRCode;                        //mX4
21199: 489 unit uPSI_BlockSocket;                   //ExtPascal
21200: 490 unit uPSI_Masks,Utils                    //VCL
21201: 491 unit uPSI_synautil;                      //Synapse!
21202: 492 unit uPSI_JclMath_Class;                 //JCL RTL
21203: 493 unit ugamdist; //Gamma function         //DMath
21204: 494 unit uibeta, ucorrel; //IBeta           //DMath
21205: 495 unit uPSI_SRMgr;                         //mX4
21206: 496 unit uPSI_HotLog;                        //mX4
21207: 497 unit uPSI_DebugBox;                      //mX4
21208: 498 unit uPSI_ustrings;                      //DMath
21209: 499 unit uPSI_uregtest;                      //DMath
21210: 500 unit uPSI_usimplex;                      //DMath
21211: 501 unit uPSI_uhyper;                        //DMath
21212: 502 unit uPSI_IdHL7;                         //Indy
21213: 503 unit uPSI_IdIPMCastBase,                 //Indy
21214: 504 unit uPSI_IdIPMCastServer;               //Indy
21215: 505 unit uPSI_IdIPMCastClient;               //Indy
21216: 506 unit uPSI_unlfit; //nlregression        //DMath
21217: 507 unit uPSI_IdRawHeaders;                  //Indy
21218: 508 unit uPSI_IdRawClient;                   //Indy
21219: 509 unit uPSI_IdRawFunctions;                //Indy
21220: 510 unit uPSI_IdTCPStream;                   //Indy
21221: 511 unit uPSI_IdSNPP;                        //Indy
21222: 512 unit uPSI_St2DBarC;                      //SysTools
21223: 513 unit uPSI_ImageWin;  //FTL              //VCL
21224: 514 unit uPSI_CustomDrawTreeView; //FTL     //VCL
21225: 515 unit uPSI_GraphWin;  //FTL              //VCL
21226: 516 unit uPSI_actionMain;  //FTL            //VCL
21227: 517 unit uPSI_StSpawn;                       //SysTools
21228: 518 unit uPSI_CtlPanel;                      //VCL
21229: 519 unit uPSI_IdLPR;                         //Indy
21230: 520 unit uPSI_SockRequestInterpreter;        //Indy
21231: 521 unit uPSI_ulambert;                      //DMath
21232: 522 unit uPSI_ucholesk;                      //DMath
21233: 523 unit uPSI_SimpleDS;                      //VCL
21234: 524 unit uPSI_DBXSqlScanner;                 //VCL
21235: 525 unit uPSI_DBXMetaDataUtil;               //VCL
21236: 526 unit uPSI_Chart;                         //TEE
21237: 527 unit uPSI_TeeProcs;                      //TEE
21238: 528 unit mXBDEUtils;                         //mX4
21239: 529 unit uPSI_MDIEdit;                       //VCL
```

```
21240: 530 unit uPSI_CopyPrsr;                    //VCL
21241: 531 unit uPSI_SockApp;                     //VCL
21242: 532 unit uPSI_AppEvnts;                    //VCL
21243: 533 unit uPSI_ExtActns;                    //VCL
21244: 534 unit uPSI_TeEngine;                    //TEE
21245: 535 unit uPSI_CoolMain; //browser          //VCL
21246: 536 unit uPSI_StCRC;                       //SysTools
21247: 537 unit uPSI_StDecMth2;                   //SysTools
21248: 538 unit uPSI_frmExportMain;               //Synedit
21249: 539 unit uPSI_SynDBEdit;                   //Synedit
21250: 540 unit uPSI_SynEditWildcardSearch;       //Synedit
21251: 541 unit uPSI_BoldComUtils;                //BOLD
21252: 542 unit uPSI_BoldIsoDateTime;             //BOLD
21253: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
21254: 544 unit uPSI_BoldXMLRequests;             //BOLD
21255: 545 unit uPSI_BoldStringList;              //BOLD
21256: 546 unit uPSI_BoldFileHandler;             //BOLD
21257: 547 unit uPSI_BoldContainers;              //BOLD
21258: 548 unit uPSI_BoldQueryUserDlg;            //BOLD
21259: 549 unit uPSI_BoldWinINet;                 //BOLD
21260: 550 unit uPSI_BoldQueue;                   //BOLD
21261: 551 unit uPSI_JvPcx;                       //JCL
21262: 552 unit uPSI_IdWhois;                     //Indy
21263: 553 unit uPSI_IdWhoIsServer;               //Indy
21264: 554 unit uPSI_IdGopher;                    //Indy
21265: 555 unit uPSI_IdDateTimeStamp;             //Indy
21266: 556 unit uPSI_IdDayTimeServer;             //Indy
21267: 557 unit uPSI_IdDayTimeUDP;                //Indy
21268: 558 unit uPSI_IdDayTimeUDPServer;          //Indy
21269: 559 unit uPSI_IdDICTServer;                //Indy
21270: 560 unit uPSI_IdDiscardServer;             //Indy
21271: 561 unit uPSI_IdDiscardUDPServer;          //Indy
21272: 562 unit uPSI_IdMappedFTP;                 //Indy
21273: 563 unit uPSI_IdMappedPortTCP;             //Indy
21274: 564 unit uPSI_IdGopherServer;              //Indy
21275: 565 unit uPSI_IdQotdServer;                //Indy
21276: 566 unit uPSI_JvRgbToHtml;                 //JCL
21277: 567 unit uPSI_JvRemLog,                    //JCL
21278: 568 unit uPSI_JvSysComp;                   //JCL
21279: 569 unit uPSI_JvTMTL;                      //JCL
21280: 570 unit uPSI_JvWinampAPI;                 //JCL
21281: 571 unit uPSI_MSysUtils;                   //mX4
21282: 572 unit uPSI_ESBMaths;                    //ESB
21283: 573 unit uPSI_ESBMaths2;                   //ESB
21284: 574 unit uPSI_uLkJSON;                     //Lk
21285: 575 unit uPSI_ZURL;  //Zeos                //Zeos
21286: 576 unit uPSI_ZSysUtils;                   //Zeos
21287: 577 unit unaUtils internals                //UNA
21288: 578 unit uPSI_ZMatchPattern;               //Zeos
21289: 579 unit uPSI_ZClasses;                    //Zeos
21290: 580 unit uPSI_ZCollections;                //Zeos
21291: 581 unit uPSI_ZEncoding;                   //Zeos
21292: 582 unit uPSI_IdRawBase;                   //Indy
21293: 583 unit uPSI_IdNTLM;                      //Indy
21294: 584 unit uPSI_IdNNTP;                      //Indy
21295: 585 unit uPSI_usniffer; //PortScanForm     //mX4
21296: 586 unit uPSI_IdCoderMIME;                 //Indy
21297: 587 unit uPSI_IdCoderUUE;                  //Indy
21298: 588 unit uPSI_IdCoderXXE;                  //Indy
21299: 589 unit uPSI_IdCoder3to4;                 //Indy
21300: 590 unit uPSI_IdCookie;                    //Indy
21301: 591 unit uPSI_IdCookieManager;             //Indy
21302: 592 unit uPSI_WDosSocketUtils;             //WDos
21303: 593 unit uPSI_WDosPlcUtils;                //WDos
21304: 594 unit uPSI_WDosPorts;                   //WDos
21305: 595 unit uPSI_WDosResolvers;               //WDos
21306: 596 unit uPSI_WDosTimers;                  //WDos
21307: 597 unit uPSI_WDosPlcs;                    //WDos
21308: 598 unit uPSI_WDosPneumatics;              //WDos
21309: 599 unit uPSI_IdFingerServer;              //Indy
21310: 600 unit uPSI_IdDNSResolver;               //Indy
21311: 601 unit uPSI_IdHTTPWebBrokerBridge;       //Indy
21312: 602 unit uPSI_IdIntercept;                 //Indy
21313: 603 unit uPSI_IdIPMCastBase;               //Indy
21314: 604 unit uPSI_IdLogBase;                   //Indy
21315: 605 unit uPSI_IdIOHandlerStream;           //Indy
21316: 606 unit uPSI_IdMappedPortUDP;             //Indy
21317: 607 unit uPSI_IdQOTDUDPServer;             //Indy
21318: 608 unit uPSI_IdQOTDUDP;                   //Indy
21319: 609 unit uPSI_IdSysLog;                    //Indy
21320: 610 unit uPSI_IdSysLogServer;              //Indy
21321: 611 unit uPSI_IdSysLogMessage;             //Indy
21322: 612 unit uPSI_IdTimeServer;                //Indy
21323: 613 unit uPSI_IdTimeUDP;                   //Indy
21324: 614 unit uPSI_IdTimeUDPServer;             //Indy
21325: 615 unit uPSI_IdUserAccounts;              //Indy
21326: 616 unit uPSI_TextUtils;                   //mX4
21327: 617 unit uPSI_MandelbrotEngine;            //mX4
21328: 618 unit uPSI_delphi_arduino_Unit1;        //mX4
```

```
21329: 619 unit uPSI_DTDSchema2;                    //mX4
21330: 620 unit uPSI_fplotMain;                     //DMath
21331: 621 unit uPSI_FindFileIter;                  //mX4
21332: 622 unit uPSI_PppState;  (JclStrHashMap)     //PPP
21333: 623 unit uPSI_PppParser;                     //PPP
21334: 624 unit uPSI_PppLexer;                      //PPP
21335: 625 unit uPSI_PCharUtils;                    //PPP
21336: 626 unit uPSI_uJSON;                         //WU
21337: 627 unit uPSI_JclStrHashMap;                 //JCL
21338: 628 unit uPSI_JclHookExcept;                 //JCL
21339: 629 unit uPSI_EncdDecd;                      //VCL
21340: 630 unit uPSI_SockAppReg;                    //VCL
21341: 631 unit uPSI_PJFileHandle;                  //PJ
21342: 632 unit uPSI_PJEnvVars;                     //PJ
21343: 633 unit uPSI_PJPipe;                        //PJ
21344: 634 unit uPSI_PJPipeFilters;                 //PJ
21345: 635 unit uPSI_PJConsoleApp;                  //PJ
21346: 636 unit uPSI_UConsoleAppEx;                 //PJ
21347: 637 unit uPSI_DbxSocketChannelNative,        //VCL
21348: 638 unit uPSI_DbxDataGenerator,              //VCL
21349: 639 unit uPSI_DBXClient;                     //VCL
21350: 640 unit uPSI_IdLogEvent;                    //Indy
21351: 641 unit uPSI_Reversi;                       //mX4
21352: 642 unit uPSI_Geometry;                      //mX4
21353: 643 unit uPSI_IdSMTPServer;                  //Indy
21354: 644 unit uPSI_Textures;                      //mX4
21355: 645 unit uPSI_IBX;                           //VCL
21356: 646 unit uPSI_IWDBCommon;                    //VCL
21357: 647 unit uPSI_SortGrid;                      //mX4
21358: 648 unit uPSI_IB;                            //VCL
21359: 649 unit uPSI_IBScript;                      //VCL
21360: 650 unit uPSI_JvCSVBaseControls;             //JCL
21361: 651 unit uPSI_Jvg3DColors;                   //JCL
21362: 652 unit uPSI_JvHLEditor;  //beat            //JCL
21363: 653 unit uPSI_JvShellHook;                   //JCL
21364: 654 unit uPSI_DBCommon2                      //VCL
21365: 655 unit uPSI_JvSHFileOperation;             //JCL
21366: 656 unit uPSI_uFilexport;                    //mX4
21367: 657 unit uPSI_JvDialogs;                     //JCL
21368: 658 unit uPSI_JvDBTreeView;                  //JCL
21369: 659 unit uPSI_JvDBUltimGrid;                 //JCL
21370: 660 unit uPSI_JvDBQueryParamsForm;           //JCL
21371: 661 unit uPSI_JvExControls;                  //JCL
21372: 662 unit uPSI_JvBDEMemTable;                 //JCL
21373: 663 unit uPSI_JvCommStatus;                  //JCL
21374: 664 unit uPSI_JvMailSlots2;                  //JCL
21375: 665 unit uPSI_JvgWinMask;                    //JCL
21376: 666 unit uPSI_StEclpse;                      //SysTools
21377: 667 unit uPSI_StMime;                        //SysTools
21378: 668 unit uPSI_StList;                        //SysTools
21379: 669 unit uPSI_StMerge;                       //SysTools
21380: 670 unit uPSI_StStrS;                        //SysTools
21381: 671 unit uPSI_StTree,                        //SysTools
21382: 672 unit uPSI_StVArr;                        //SysTools
21383: 673 unit uPSI_StRegIni;                      //SysTools
21384: 674 unit uPSI_urkf;                          //DMath
21385: 675 unit uPSI_usvd;                          //DMath
21386: 676 unit uPSI_DepWalkUtils;                  //JCL
21387: 677 unit uPSI_OptionsFrm;                    //JCL
21388: 678 unit yuvconverts;                        //mX4
21389: 679 uPSI_JvPropAutoSave;                     //JCL
21390: 680 uPSI_AclAPI;                             //alcinoe
21391: 681 uPSI_AviCap;                             //alcinoe
21392: 682 uPSI_ALAVLBinaryTree;                    //alcinoe
21393: 683 uPSI_ALFcnMisc;                          //alcinoe
21394: 684 uPSI_ALStringList;                       //alcinoe
21395: 685 uPSI_ALQuickSortList;                    //alcinoe
21396: 686 uPSI_ALStaticText;                       //alcinoe
21397: 687 uPSI_ALJSONDoc;                          //alcinoe
21398: 688 uPSI_ALGSMComm;                          //alcinoe
21399: 689 uPSI_ALWindows;                          //alcinoe
21400: 690 uPSI_ALMultiPartFormDataParser;          //alcinoe
21401: 691 uPSI_ALHttpCommon;                       //alcinoe
21402: 692 uPSI_ALWebSpider,                        //alcinoe
21403: 693 uPSI_ALHttpClient;                       //alcinoe
21404: 694 uPSI_ALFcnHTML;                          //alcinoe
21405: 695 uPSI_ALFTPClient;                        //alcinoe
21406: 696 uPSI_ALInternetMessageCommon;            //alcinoe
21407: 697 uPSI_ALWininetHttpClient;                //alcinoe
21408: 698 uPSI_ALWinInetFTPClient;                 //alcinoe
21409: 699 uPSI_ALWinHttpWrapper;                   //alcinoe
21410: 700 uPSI_ALWinHttpClient;                    //alcinoe
21411: 701 uPSI_ALFcnWinSock;                       //alcinoe
21412: 702 uPSI_ALFcnSQL;                           //alcinoe
21413: 703 uPSI_ALFcnCGI;                           //alcinoe
21414: 704 uPSI_ALFcnExecute;                       //alcinoe
21415: 705 uPSI_ALFcnFile;                          //alcinoe
21416: 706 uPSI_ALFcnMime;                          //alcinoe
21417: 707 uPSI_ALPhpRunner;                        //alcinoe
```

```
21418: 708 uPSI_ALGraphic;                          //alcinoe
21419: 709 uPSI_ALIniFiles;                         //alcinoe
21420: 710 uPSI_ALMemCachedClient;                  //alcinoe
21421: 711 unit uPSI_MyGrids;                       //mX4
21422: 712 uPSI_ALMultiPartMixedParser              //alcinoe
21423: 713 uPSI_ALSMTPClient                        //alcinoe
21424: 714 uPSI_ALNNTPClient;                       //alcinoe
21425: 715 uPSI_ALHintBalloon;                      //alcinoe
21426: 716 unit uPSI_ALXmlDoc;                      //alcinoe
21427: 717 unit uPSI_IPCThrd;                       //VCL
21428: 718 unit uPSI_MonForm;                       //VCL
21429: 719 unit uPSI_TeCanvas;                      //Orpheus
21430: 720 unit uPSI_Ovcmisc;                       //Orpheus
21431: 721 unit uPSI_ovcfiler;                      //Orpheus
21432: 722 unit uPSI_ovcstate;                      //Orpheus
21433: 723 unit uPSI_ovccoco;                       //Orpheus
21434: 724 unit uPSI_ovcrvexp;                      //Orpheus
21435: 725 unit uPSI_OvcFormatSettings;             //Orpheus
21436: 726 unit uPSI_OvcUtils;                      //Orpheus
21437: 727 unit uPSI_ovcstore;                      //Orpheus
21438: 728 unit uPSI_ovcstr;                        //Orpheus
21439: 729 unit uPSI_ovcmru;                        //Orpheus
21440: 730 unit uPSI_ovccmd;                        //Orpheus
21441: 731 unit uPSI_ovctimer;                      //Orpheus
21442: 732 unit uPSI_ovcintl;                       //Orpheus
21443: 733 uPSI_AfCircularBuffer;                   //AsyncFree
21444: 734 uPSI_AfUtils;                           //AsyncFree
21445: 735 uPSI_AfSafeSync;                        //AsyncFree
21446: 736 uPSI_AfComPortCore;                     //AsyncFree
21447: 737 uPSI_AfComPort;                         //AsyncFree
21448: 738 uPSI_AfPortControls;                    //AsyncFree
21449: 739 uPSI_AfDataDispatcher;                  //AsyncFree
21450: 740 uPSI_AfViewers;                         //AsyncFree
21451: 741 uPSI_AfDataTerminal;                    //AsyncFree
21452: 742 uPSI_SimplePortMain;                    //AsyncFree
21453: 743 unit uPSI_ovcclock;                      //Orpheus
21454: 744 unit uPSI_o32intlst;                     //Orpheus
21455: 745 unit uPSI_o32ledlabel;                   //Orpheus
21456: 746 unit uPSI_AlMySqlClient;                 //alcinoe
21457: 747 unit uPSI_ALFBXClient;                   //alcinoe
21458: 748 unit uPSI_ALFcnSQL;                      //alcinoe
21459: 749 unit uPSI_AsyncTimer;                    //mX4
21460: 750 unit uPSI_ApplicationFileIO;             //mX4
21461: 751 unit uPSI_PsAPI;                         //VCLé
21462: 752 uPSI_ovcuser;                           //Orpheus
21463: 753 uPSI_ovcurl;                            //Orpheus
21464: 754 uPSI_ovcvlb;                            //Orpheus
21465: 755 uPSI_ovccolor;                          //Orpheus
21466: 756 uPSI_ALFBXLib,                          //alcinoe
21467: 757 uPSI_ovcmeter;                          //Orpheus
21468: 758 uPSI_ovcpeakm;                          //Orpheus
21469: 759 uPSI_O32BGSty;                          //Orpheus
21470: 760 uPSI_ovcBidi;                           //Orpheus
21471: 761 uPSI_ovctcary;                          //Orpheus
21472: 762 uPSI_DXPUtils;                          //mX4
21473: 763 uPSI_ALMultiPartBaseParser;             //alcinoe
21474: 764 uPSI_ALMultiPartAlternativeParser;      //alcinoe
21475: 765 uPSI_ALPOP3Client;                      //alcinoe
21476: 766 uPSI_SmallUtils;                        //mX4
21477: 767 uPSI_MakeApp;                           //mX4
21478: 768 uPSI_O32MouseMon;                       //Orpheus
21479: 769 uPSI_OvcCache;                          //Orpheus
21480: 770 uPSI_ovccalc;                           //Orpheus
21481: 771 uPSI_Joystick,                          //OpenGL
21482: 772 uPSI_ScreenSaver;                       //OpenGL
21483: 773 uPSI_XCollection,                       //OpenGL
21484: 774 uPSI_Polynomials,                       //OpenGL
21485: 775 uPSI_PersistentClasses, //9.86          //OpenGL
21486: 776 uPSI_VectorLists;                       //OpenGL
21487: 777 uPSI_XOpenGL,                          //OpenGL
21488: 778 uPSI_MeshUtils;                         //OpenGL
21489: 779 unit uPSI_JclSysUtils;                  //JCL
21490: 780 unit uPSI_JclBorlandTools;              //JCL
21491: 781 unit JclFileUtils_max;                  //JCL
21492: 782 uPSI_AfDataControls,                    //AsyncFree
21493: 783 uPSI_GLSilhouette;                      //OpenGL
21494: 784 uPSI_JclSysUtils_class;                 //JCL
21495: 785 uPSI_JclFileUtils_class;                //JCL
21496: 786 uPSI_FileUtil;                          //JCL
21497: 787 uPSI_changefind;                        //mX4
21498: 788 uPSI_cmdIntf;                           //mX4
21499: 789 uPSI_fservice;                          //mX4
21500: 790 uPSI_Keyboard;                          //OpenGL
21501: 791 uPSI_VRMLParser,                        //OpenGL
21502: 792 uPSI_GLFileVRML,                        //OpenGL
21503: 793 uPSI_Octree;                            //OpenGL
21504: 794 uPSI_GLPolyhedron,                      //OpenGL
21505: 795 uPSI_GLCrossPlatform;                   //OpenGL
21506: 796 uPSI_GLParticles;                       //OpenGL
```

```
21507: 797 uPSI_GLNavigator;                    //OpenGL
21508: 798 uPSI_GLStarRecord;                   //OpenGL
21509: 799 uPSI_GLTextureCombiners;             //OpenGL
21510: 800 uPSI_GLCanvas;                       //OpenGL
21511: 801 uPSI_GeometryBB;                     //OpenGL
21512: 802 uPSI_GeometryCoordinates;           //OpenGL
21513: 803 uPSI_VectorGeometry;                 //OpenGL
21514: 804 uPSI_BumpMapping;                    //OpenGL
21515: 805 uPSI_TGA;                           //OpenGL
21516: 806 uPSI_GLVectorFileObjects;           //OpenGL
21517: 807 uPSI_IMM;                           //VCL
21518: 808 uPSI_CategoryButtons;               //VCL
21519: 809 uPSI_ButtonGroup;                   //VCL
21520: 810 uPSI_DbExcept;                      //VCL
21521: 811 uPSI_AxCtrls;                       //VCL
21522: 812 uPSI_GL_actorUnit1;                 //OpenGL
21523: 813 uPSI_StdVCL;                        //VCL
21524: 814 unit CurvesAndSurfaces;             //OpenGL
21525: 815 uPSI_DataAwareMain;                 //AsyncFree
21526: 816 uPSI_TabNotBk;                      //VCL
21527: 817 uPSI_udwsfiler;                     //mX4
21528: 818 uPSI_synaip;                        //Synapse!
21529: 819 uPSI_synacode;                      //Synapse
21530: 820 uPSI_synachar;                      //Synapse
21531: 821 uPSI_synamisc;                      //Synapse
21532: 822 uPSI_synaser;                       //Synapse
21533: 823 uPSI_synaicnv;                      //Synapse
21534: 824 uPSI_tlntsend;                      //Synapse
21535: 825 uPSI_pingsend;                      //Synapse
21536: 826 uPSI_blcksock;                      //Synapse
21537: 827 uPSI_asn1util;                      //Synapse
21538: 828 uPSI_dnssend;                       //Synapse
21539: 829 uPSI_clamsend;                      //Synapse
21540: 830 uPSI_ldapsend;                      //Synapse
21541: 831 uPSI_mimemess;                      //Synapse
21542: 832 uPSI_slogsend;                      //Synapse
21543: 833 uPSI_mimepart;                      //Synapse
21544: 834 uPSI_mimeinln;                      //Synapse
21545: 835 uPSI_ftpsend,                       //Synapse
21546: 836 uPSI_ftptsend;                      //Synapse
21547: 837 uPSI_httpsend;                      //Synapse
21548: 838 uPSI_sntpsend;                      //Synapse
21549: 839 uPSI_smtpsend;                      //Synapse
21550: 840 uPSI_snmpsend;                      //Synapse
21551: 841 uPSI_imapsend;                      //Synapse
21552: 842 uPSI_pop3send;                      //Synapse
21553: 843 uPSI_nntpsend;                      //Synapse
21554: 844 uPSI_ssl_cryptlib;                  //Synapse
21555: 845 uPSI_ssl_openssl;                   //Synapse
21556: 846 uPSI_synhttp_daemon;                //Synapse
21557: 847 uPSI_NetWork;                       //mX4
21558: 848 uPSI_PingThread;                    //Synapse
21559: 849 uPSI_JvThreadTimer;                 //JCL
21560: 850 unit uPSI_wwSystem;                 //InfoPower
21561: 851 unit uPSI_IdComponent;              //Indy
21562: 852 unit uPSI_IdIOHandlerThrottle;      //Indy
21563: 853 unit uPSI_Themes;                   //VCL
21564: 854 unit uPSI_StdStyleActnCtrls;        //VCL
21565: 855 unit uPSI_UDDIHelper;               //VCL
21566: 856 unit uPSI_IdIMAP4Server;            //Indy
21567: 857 uPSI_VariantSymbolTable,            //VCL //3.9.9.92
21568: 858 uPSI_udf_glob,                      //mX4
21569: 859 uPSI_TabGrid,                       //VCL
21570: 860 uPSI_JsDBTreeView,                  //mX4
21571: 861 uPSI_JsSendMail,                    //mX4
21572: 862 uPSI_dbTvRecordList,                //mX4
21573: 863 uPSI_TreeVwEx,                      //mX4
21574: 864 uPSI_ECDataLink,                    //mX4
21575: 865 uPSI_dbTree,                        //mX4
21576: 866 uPSI_dbTreeCBox,                    //mX4
21577: 867 unit uPSI_Debug; //TfrmDebug       //mX4
21578: 868 uPSI_TimeFncs;                      //mX4
21579: 869 uPSI_FileIntf,                      //VCL
21580: 870 uPSI_SockTransport,                 //RTL
21581: 871 unit uPSI_WinInet;                  //RTL
21582: 872 unit uPSI_Wwstr;                    //mX4
21583: 873 uPSI_DBLookup,                      //VCL
21584: 874 uPSI_Hotspot,                       //mX4
21585: 875 uPSI_HList; //History List          //mX4
21586: 876 unit uPSI_DrTable;                  //VCL
21587: 877 uPSI_TConnect,                      //VCL
21588: 978 uPSI_DataBkr,                       //VCL
21589: 979 uPSI_HTTPIntr;                      //VCL
21590: 980 unit uPSI_Mathbox;                  //mX4
21591: 881 uPSI_cyIndy,                        //cY
21592: 882 uPSI_cySysUtils,                    //cY
21593: 883 uPSI_cyWinUtils,                    //cY
21594: 884 uPSI_cyStrUtils,                    //cY
21595: 885 uPSI_cyObjUtils,                    //cY
```

```
21596: 886 uPSI_cyDateUtils,                          //cY
21597: 887 uPSI_cyBDE,                                //cY
21598: 888 uPSI_cyClasses,                            //cY
21599: 889 uPSI_cyGraphics,  //3.9.9.94_2             //cY
21600: 890 unit uPSI_cyTypes;                         //cY
21601: 891 uPSI_JvDateTimePicker,                     //JCL
21602: 892 uPSI_JvCreateProcess,                      //JCL
21603: 893 uPSI_JvEasterEgg,                          //JCL
21604: 894 uPSI_WinSvc,  //3.9.9.94_3                 //VCL
21605: 895 uPSI_SvcMgr                                //VCL
21606: 896 unit uPSI_JvPickDate;                      //JCL
21607: 897 unit uPSI_JvNotify;                        //JCL
21608: 898 uPSI_JvStrHlder                            //JCL
21609: 899 unit uPSI_JclNTFS2;                        //JCL
21610: 900 uPSI_Jcl8087 //math coprocessor           //JCL
21611: 901 uPSI_JvAddPrinter                          //JCL
21612: 902 uPSI_JvCabFile                             //JCL
21613: 903 uPSI_JvDataEmbedded;                       //JCL
21614: 904 unit uPSI_U_HexView;                       //mX4
21615: 905 uPSI_UWavein4,                             //mX4
21616: 906 uPSI_AMixer,                               //mX4
21617: 907 uPSI_JvaScrollText,                        //mX4
21618: 908 uPSI_JvArrow,                             //mX4
21619: 909 unit uPSI_UrlMon;                         //mX4
21620: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'  //mX4
21621: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain; //DFF
21622: 912 unit uPSI_DFFUtils;                        //DFF
21623: 913 unit uPSI_MathsLib;                       //DFF
21624: 914 uPSI_UIntList;                            //DFF
21625: 915 uPSI_UGetParens;                          //DFF
21626: 916 unit uPSI_UGeometry;                      //DFF
21627: 917 unit uPSI_UAstronomy;                     //DFF
21628: 918 unit uPSI_UCardComponentV2;               //DFF
21629: 919 unit uPSI_UTGraphSearch;                  //DFF
21630: 920 unit uPSI_UParser10;                      //DFF
21631: 921 unit uPSI_cyIEUtils;                      //cY
21632: 922 unit uPSI_UcomboV2;                       //DFF
21633: 923 uPSI_cyBaseComm,                          //cY
21634: 924 uPSI_cyAppInstances,                      //cY
21635: 925 uPSI_cyAttract,                           //cY
21636: 926 uPSI_cyDERUtils                           //cY
21637: 927 unit uPSI_cyDocER;                        //cY
21638: 928 unit uPSI_ODBC;                           //mX
21639: 929 unit uPSI_AssocExec;                      //mX
21640: 930 uPSI_cyBaseCommRoomConnector,             //cY
21641: 931 uPSI_cyCommRoomConnector,                 //cY
21642: 932 uPSI_cyCommunicate,                       //cY
21643: 933 uPSI_cyImage;                             //cY
21644: 934 uPSI_cyBaseContainer                      //cY
21645: 935 uPSI_cyModalContainer,                    //cY
21646: 936 uPSI_cyFlyingContainer;                   //cY
21647: 937 uPSI_RegStr,                              //VCL
21648: 938 uPSI_HtmlHelpViewer;                      //VCL
21649: 939 unit uPSI_cyIniForm                       //cY
21650: 940 unit uPSI_cyVirtualGrid;                  //cY
21651: 941 uPSI_Profiler,                            //DA
21652: 942 uPSI_BackgroundWorker,                    //DA
21653: 943 uPSI_WavePlay,                            //DA
21654: 944 uPSI_WaveTimer,                           //DA
21655: 945 uPSI_WaveUtils;                           //DA
21656: 946 uPSI_NamedPipes,                          //TB
21657: 947 uPSI_NamedPipeServer,                     //TB
21658: 948 unit uPSI_process,                        //TB
21659: 949 unit uPSI_DPUtils;                        //TB
21660: 950 unit uPSI_CommonTools;                    //TB
21661: 951 uPSI_DataSendToWeb,                       //TB
21662: 952 uPSI_StarCalc,                            //TB
21663: 953 uPSI_D2_XPVistaHelperU                    //TB
21664: 954 unit uPSI_NetTools                        //TB
21665: 955 unit uPSI_Pipes                           //TB
21666: 956 uPSI_ProcessUnit,                        //mX
21667: 957 uPSI_adGSM,                               //mX
21668: 958 unit uPSI_BetterADODataSet;               //mX
21669: 959 unit uPSI_AdSelCom; //FTT                 //mX
21670: 960 unit unit uPSI_dwsXPlatform;              //DWS
21671: 961 uPSI_AdSocket;                            //mX Turbo Power
21672: 962 uPSI_AdPacket;                            //mX
21673: 963 uPSI_AdPort;                              //mX
21674: 964 uPSI_PathFunc;                            //Inno
21675: 965 uPSI_CmnFunc;                             //Inno
21676: 966 uPSI_CmnFunc2; //Inno Setup               //Inno
21677: 967 unit uPSI_BitmapImage;                    //mX4
21678: 968 unit uPSI_ImageGrabber;                   //mX4
21679: 969 uPSI_SecurityFunc,                        //Inno
21680: 970 uPSI_RedirFunc,                           //Inno
21681: 971 uPSI_FIFO, (MemoryStream)                 //mX4
21682: 972 uPSI_Int64Em,                             //Inno
21683: 973 unit uPSI_InstFunc;                       //Inno
21684: 974 unit uPSI_LibFusion;                      //Inno
```

```
21685: 975 uPSI_SimpleExpression;                    //Inno
21686: 976 uPSI_unitResourceDetails,                 //XN
21687: 977 uPSI_unitResFile,                         //XN
21688: 978 unit uPSI_simpleComport;                  //mX4
21689: 979 unit uPSI_AfViewershelpers;               //Async
21690: 980 unit uPSI_Console;                        //mX4
21691: 981 unit uPSI_AnalogMeter;                    //TB
21692: 982 unit uPSI_XPrinter,                       //TB
21693:
21694:
21695:
21696:
21697: ////////////////////////////////////////////////////////////////////////////
21698: //Form Template Library FTL
21699: ////////////////////////////////////////////////////////////////////////////
21700:
21701: 32 FTL For Form Building out of the Script, eg. 399_form_templates.txt
21702:
21703: 045 unit uPSI_VListView                       TFormListView;
21704: 263 unit uPSI_JvProfiler32;                   TProfReport
21705: 270 unit uPSI_ImgList;                        TCustomImageList
21706: 278 unit uPSI_JvImageWindow;                  TJvImageWindow
21707: 317 unit uPSI_JvParserForm;                   TJvHTMLParserForm
21708: 497 unit uPSI_DebugBox;                       TDebugBox
21709: 513 unit uPSI_ImageWin;                       TImageForm, TImageForm2
21710: 514 unit uPSI_CustomDrawTreeView;             TCustomDrawForm
21711: 515 unit uPSI_GraphWin;                       TGraphWinForm
21712: 516 unit uPSI_actionMain;                     TActionForm
21713: 518 unit uPSI_CtlPanel;                       TAppletApplication
21714: 529 unit uPSI_MDIEdit;                       TEditForm
21715: 535 unit uPSI_CoolMain; {browser}            TWebMainForm
21716: 538 unit uPSI_frmExportMain;                 TSynexportForm
21717: 585 unit uPSI_usniffer; {//PortScanForm}     TSniffForm
21718: 600 unit uPSI_ThreadForm;                    TThreadSortForm;
21719: 618 unit uPSI_delphi_arduino_Unit1;          TLEDForm
21720: 620 unit uPSI_fplotMain;                     TfplotForm1
21721: 660 unit uPSI_JvDBQueryParamsForm;           TJvQueryParamsDialog
21722: 677 unit uPSI_OptionsFrm;                    TfrmOptions;
21723: 718 unit uPSI_MonForm;                       TMonitorForm
21724: 742 unit uPSI_SimplePortMain;               TPortForm1
21725: 770 unit uPSI_ovccalc;                      TOvcCalculator  //widget
21726: 810 unit uPSI_DbExcept;                     TDbEngineErrorDlg
21727: 812 unit uPSI_GL_actorUnit1;                TglActorForm1  //OpenGL Robot
21728: 846 unit uPSI_synhttp_daemon;               TTCPHttpDaemon, TTCPHttpThrd, TPingThread
21729: 867 unit uPSI_Debug;                        TfrmDebug
21730: 904 unit uPSI_U_HexView;                    THexForm2
21731: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)  TOscfrmMain
21732: 959 unit uPSI_AdSelCom;                     TComSelectForm
21733:
21734:
21735: ex.:with TEditForm.create(self) do begin
21736:       caption:= 'Template Form Tester';
21737:       FormStyle:= fsStayOnTop;
21738:       with editor do begin
21739:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
21740:         SelStart:= 0;
21741:         Modified:= False;
21742:       end;
21743:     end;
21744:   with TWebMainForm.create(self) do begin
21745:     URLs.Text:= 'http://www.kleiner.ch';
21746:     URLsClick(self); Show;
21747:   end;
21748:   with TSynexportForm.create(self) do begin
21749:     Caption:= 'Synexport HTML RTF tester';
21750:     Show;
21751:   end;
21752:   with TThreadSortForm.create(self) do begin
21753:      showmodal; free;
21754:   end;
21755:   with TCustomDrawForm.create(self) do begin
21756:        width:=820; height:=820;
21757:        image1.height:= 600; //add properties
21758:        image1.picture.bitmap:= image2.picture.bitmap;
21759:        //SelectionBackground1Click(self) CustomDraw1Click(self);
21760:        Background1.click;
21761:        bitmap1.click;
21762:        Tile1.click;
21763:        Showmodal;
21764:        Free;
21765:      end;
21766:    with TfplotForm1.Create(self) do begin
21767:      BtnPlotClick(self);
21768:      Showmodal; Free;
21769:    end;
21770:   with TOvcCalculator.create(self) do begin
21771:     parent:= aForm;
21772:    //free;
21773:     setbounds(550,510,200,150);
```

```
21774:     displaystr:= 'maXcalc';
21775:   end;
21776:   with THexForm2.Create(self) do begin
21777:      ShowModal;
21778:      Free;
21779:   end;
21780:
21781: function CheckBox: string;
21782: var idHTTP: TIDHTTP;
21783:  begin
21784:    result:= 'version not found';
21785:    if IsInternet then begin
21786:      idHTTP:= TIdHTTP.Create(NIL);
21787:      try
21788:        result:= idHTTP.Get(MXVERSIONFILE2);
21789:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
21790:        if result = MBVER2 then begin
21791:          //output.Font.Style:= [fsbold];
21792:          //Speak(' A new Version '+vstr+' of max box is available! ');
21793:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
21794:        end;
21795:       //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
21796:      finally
21797:        idHTTP.Free
21798:    end;
21799:  end;
21800: end;
21801:
21802:
21803: ////////////////////////////////////////////////////////////////////////////
21804: All maXbox Tutorials Table of Content 2014
21805: ////////////////////////////////////////////////////////////////////////////
21806:     Tutorial 00 Function-Coding  (Blix the Programmer)
21807:     Tutorial 01 Procedural-Coding
21808:     Tutorial 02 OO-Programming
21809:     Tutorial 03 Modular Coding
21810:     Tutorial 04 UML Use Case Coding
21811:     Tutorial 05 Internet Coding
21812:     Tutorial 06 Network Coding
21813:     Tutorial 07 Game Graphics Coding
21814:     Tutorial 08 Operating System Coding
21815:     Tutorial 09 Database Coding
21816:     Tutorial 10 Statistic Coding
21817:     Tutorial 11 Forms Coding
21818:     Tutorial 12 SQL DB Coding
21819:     Tutorial 13 Crypto Coding
21820:     Tutorial 14 Parallel Coding
21821:     Tutorial 15 Serial RS232 Coding
21822:     Tutorial 16 Event Driven Coding
21823:     Tutorial 17 Web Server Coding
21824:     Tutorial 18 Arduino System Coding
21825:     Tutorial 18_3 RGB LED System Coding
21826:     Tutorial 19 WinCOM /Arduino Coding
21827:     Tutorial 20 Regular Expressions RegEx
21828:     Tutorial 21 Android Coding (coming 2013)
21829:     Tutorial 22 Services Programming
21830:     Tutorial 23 Real Time Systems
21831:     Tutorial 24 Clean Code
21832:     Tutorial 25 maXbox Configuration I+II
21833:     Tutorial 26 Socket Programming with TCP
21834:     Tutorial 27 XML & TreeView
21835:     Tutorial 28 DLL Coding (available)
21836:     Tutorial 29 UML Scripting (2014)
21837:     Tutorial 30 Web of Things (2014)
21838:     Tutorial 31 Closures (coming 2014)
21839:     Tutorial 32 SQL Firebird (coming 2014)
21840:     Tutorial 33 Oscilloscope (coming 2015)
21841:     Tutorial 34 GPS Navigation (coming 2015)
21842:     Tutorial 35 Web Box (available)
21843:
21844: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
21845: using Docu for this file is maxbox_functions_all.pdf
21846: PEP - Pascal Education Program  Lib Lab ShellHell
21847:
21848: http://stackoverflow.com/tags/pascalscript/hot
21849: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
21850: http://sourceforge.net/projects/maXbox   #locs:51620
21851: http://sourceforge.net/apps/mediawiki/maXbox
21852: http://www.blaisepascal.eu/
21853: https://github.com/maxkleiner/maXbox3.git
21854: http://www.heise.de/download/maxbox-1176464.html
21855: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
21856: https://www.facebook.com/pages/Programming-maXbox/166844836691703
21857: http://www.softwareschule.ch/arduino_training.pdf
21858: http://www.delphiarea.com
21859:
21860: All maXbox Examples List
21861: https://github.com/maxkleiner/maXbox3/releases
21862: ********************************************************************************************
```

21863: 000_pas_baseconvert.txt
21864: 000_pas_baseconvert.txt_encrypt
21865: 000_pas_baseconvert.txt_decrypt
21866: 001_1_pas_functest - Kopie.txt
21867: 001_1_pas_functest.txt
21868: 001_1_pas_functest2.txt
21869: 001_1_pas_functest_clx2.txt
21870: 001_1_pas_functest_clx2_2.txt
21871: 001_1_pas_functest_openarray.txt
21872: 001_pas_lottogen.txt
21873: 001_pas_lottogen_template.txt
21874: 001_pas_lottogen.txtcopy
21875: 002_pas_russianroulette.txt
21876: 002_pas_russianroulette.txtcopy
21877: 002_pas_russianroulette.txtcopy_decrypt
21878: 002_pas_russianroulette.txtcopy_encrypt
21879: 003_pas_motion.txt
21880: 003_pas_motion.txtcopy
21881: 004_pas_search.txt
21882: 004_pas_search_replace.txt
21883: 004_search_replace_allfunctionlist.txt
21884: 005_pas_oodesign.txt
21885: 005_pas_shelllink.txt
21886: 006_pas_oobatch.txt
21887: 007_pas_streamcopy.txt
21888: 008_EINMALEINS_FUNC.TXT
21889: 008_explanation.txt
21890: 008_pas_verwechselt.txt
21891: 008_pas_verwechselt_ibz_bern_func.txt
21892: 008_stack_ibz.TXT
21893: 009_pas_umlrunner.txt
21894: 009_pas_umlrunner_all.txt
21895: 009_pas_umlrunner_componenttest.txt
21896: 009_pas_umlrunner_solution.txt
21897: 009_pas_umlrunner_solution_2step.txt
21898: 010_pas_oodesign_solution.txt
21899: 011_pas_puzzlepas_defect.txt
21900: 012_pas_umlrunner_solution.txt
21901: 012_pas_umlrunner_solution2.txt
21902: 013_pas_linenumber.txt
21903: 014_pas_primetest.txt
21904: 014_pas_primetest_first.txt
21905: 014_pas_primetest_sync.txt
21906: 015_pas_designbycontract.txt
21907: 015_pas_designbycontract_solution.txt
21908: 016_pas_searchrec.txt
21909: 017_chartgen.txt
21910: 018_data_simulator.txt
21911: 019_dez_to_bin.txt
21912: 019_dez_to_bin_grenzwert_ibz.txt
21913: 020_proc_feedback.txt
21914: 021_pas_symkey.txt
21915: 021_pas_symkey_solution.txt
21916: 022_pas_filestreams.txt
21917: 023_pas_find_searchrec.txt
21918: 023_pas_pathfind.txt
21919: 024_pas_TFileStream_records.txt
21920: 025_prime_direct.txt
21921: 026_pas_memorystream.txt
21922: 027_pas_shellexecute_beta.txt
21923: 027_pas_shellexecute_solution.txt
21924: 028_pas_dataset.txt
21925: 029_pas_assignfile.txt
21926: 029_pas_assignfile_dragndropexe.txt
21927: 030_palindrome_2.txt
21928: 030_palindrome_tester.txt
21929: 030_pas_recursion.txt
21930: 030_pas_recursion2.txt
21931: 031_pas_hashcode.txt
21932: 032_pas_crc_const.txt
21933: 033_pas_cipher.txt
21934: 033_pas_cipher_def.txt
21935: 033_pas_cipher_file_2_solution.txt
21936: 034_pas_soundbox.txt
21937: 035_pas_crcscript.txt
21938: 035_pas_CRCscript_modbus.txt
21939: 036_pas_includetest.txt
21940: 036_pas_includetest_basta.txt
21941: 037_pas_define_demo32.txt
21942: 038_pas_box_demonstrator.txt
21943: 039_pas_dllcall.txt
21944: 040_paspointer.txt
21945: 040_paspointer_old.txt
21946: 041_pasplotter.txt
21947: 041_pasplotter_plus.txt
21948: 042_pas_kgv_ggt.txt
21949: 043_pas_proceduretype.txt
21950: 044_pas_14queens_solwith14.txt
21951: 044_pas_8queens.txt

282_fadengraphik.txt
283_SQL_API_messagetimeout.txt
284_SysTools4.txt
285_MineForm_GR32.TXT
285_MineForm_GR32main.TXT
285_MineForm_GR32mainsolution.TXT
285_MineForm_propas.TXT
285_MineForm_propas2.TXT
285_minesweeper2.TXT
285_Patterns_process.txt
286_colormixer_jpeg_charcounter.txt
286_colormixer_jpeg_charcounter2.txt
287_eventhandling.txt
287_eventhandling2.txt
287_eventhandling2_negpower.txt
288_bitblt.txt
288_bitblt_resize.txt
289_regression.txt
289_regression2.txt
290_bestofbox.txt
290_bestofbox2.txt
290_bestofbox3.txt
291_3sort_visual_thread.txt
292_refactoring2.txt
293_bold_utils.txt
293_ib_utils.txt
293_ib_utils_timetest.txt
294_maxcalc_demo.txt
294_maxcalc_demo2.txt
295_easter_calendar.txt
295_easter_calendar2.txt
295_easter_combobox.txt
297_atomimage.txt
297_atomimage2.txt
297_atomimage3.txt
297_atomimage4.txt
297_maxonmotor.TXT
297_maxon_atomimage9.txt
298_bitblt_animation.txt
298_bitblt_animation2.txt
298_bitblt_animation3.txt
298_bitblt_animation4.txt
298_bitblt_animation4_screensaver.txt
298_bitblt_animation5_screensaver.txt
299_animation.txt
299_animationmotor_arduino.txt
299_animation_formprototype.txt
299_realtimeclock_arduino.txt
299_realtimeclock_arduino2.txt
300_treeview.txt
300_treeview_test.txt
300_treeview_test2.txt
300_treeview_test3.txt
301_LED_Arduino3.txt
301_led_arduino3_simple.txt
301_led_arduino3_simplecode.txt
301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt
301_SQL_DBfirebird4.txt
302_LCLActivity_java.txt
302_LED_DataLogger.txt
303_Android_LCLActivity_java.txt
303_webserver.txt
303_webserver2.txt
303_webserver_alldocs2.txt
303_webserver_alldocs2_tester.txt
303_webserver_minimal.txt
303_webserver_simple.txt
304_st_system.txt
305_indy_elizahttpserver.TXT
305_indy_elizahttpserver2.TXT
305_indy_elizahttpserver3.TXT
305_indy_elizahttpserver4file.TXT
305_webserver_arduino.txt
305_webserver_arduino2.txt
305_webserver_arduino3.txt
305_webserver_arduino3ibz.txt
305_webserver_arduino3ibz_rgb_led.txt
305_webserver_arduino3test.txt
306_SPS_http_command.txt
307_all_booleanlogic.txt
308_bitbox3.txt
308_bitbox3_exec.txt
308_boolean_animation.txt
308_boolean_animation2.txt
309_regex_power.txt
309_regex_powertester2.txt
309_regex_powertester3.txt

21952: 044_pas_8queens_sol2.txt
21953: 044_pas_8queens_solutions.txt
21954: 044_queens_performer.txt
21955: 044_queens_performer2.txt
21956: 044_queens_performer2tester.txt
21957: 045_pas_listhandling.txt
21958: 046_pas_records.txt
21959: 047_pas_modula10.txt
21960: 048_pas_romans.txt
21961: 049_pas_ifdemo.txt
21962: 049_pas_ifdemo_BROKER.txt
21963: 050_pas_primetest2.txt
21964: 050_pas_primetester_thieves.txt
21965: 050_program_starter.txt
21966: 050_program_starter_performance.txt
21967: 051_pas_findtext_solution.txt
21968: 052_pas_text_as_stream.txt
21969: 052_pas_text_as_stream_include.txt
21970: 053_pas_singleton.txt
21971: 054_pas_speakpassword.txt
21972: 054_pas_speakpassword2.txt
21973: 054_pas_speakpassword_searchtest.txt
21974: 055_pas_factorylist.txt
21975: 056_pas_demeter.txt
21976: 057_pas_dirfinder.txt
21977: 058_pas_filefinder.txt
21978: 058_pas_filefinder_pdf.txt
21979: 058_pas_filefinder_screview.txt
21980: 058_pas_filefinder_screview2.txt
21981: 058_pas_filefinder_screview3.txt
21982: 059_pas_timertest.txt
21983: 059_pas_timertest_2.txt
21984: 059_pas_timertest_time_solution.txt
21985: 059_timerobject_starter2.txt
21986: 059_timerobject_starter2_ibz2_async.txt
21987: 059_timerobject_starter2_uml.txt
21988: 059_timerobject_starter2_uml_main.txt
21989: 059_timerobject_starter4_ibz.txt
21990: 060_pas_datefind.txt
21991: 060_pas_datefind_exceptions2.txt
21992: 060_pas_datefind_exceptions_CHECKTEST.txt
21993: 060_pas_datefind_fulltext.txt
21994: 060_pas_datefind_plus.txt
21995: 060_pas_datefind_plus_mydate.txt
21996: 061_pas_randomwalk.txt
21997: 061_pas_randomwalk_plus.txt
21998: 062_paskorrelation.txt
21999: 063_pas_calculateform.txt
22000: 063_pas_calculateform_2list.txt
22001: 064_pas_timetest.txt
22002: 065_pas_bitcounter.txt
22003: 066_pas_eliza.txt
22004: 066_pas_eliza_include_sol.txt
22005: 067_pas_morse.txt
22006: 068_pas_piezo_sound.txt
22007: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
22008: 069_my_LEDBOX.TXT
22009: 069_pas_ledmatrix.txt
22010: 069_pas_LEDMATRIX_Alphabet.txt
22011: 069_pas_LEDMATRIX_Alphabet_run.txt
22012: 069_pas_LEDMATRIX_Alphabet_tester.txt
22013: 069_PAS_LEDMATRIX_COLOR.TXT
22014: 069_pas_ledmatrix_fixedit.txt
22015: 069_pas_LEDMATRIX_soundbox.txt
22016: 069_pas_LEDMATRIX_soundbox2.txt
22017: 069_Richter_MATRIX.TXT
22018: 070_pas_functionplot.txt
22019: 070_pas_functionplotter2.txt
22020: 070_pas_functionplotter2_mx4.txt
22021: 070_pas_functionplotter2_tester.txt
22022: 070_pas_functionplotter3.txt
22023: 070_pas_functionplotter4.txt
22024: 070_pas_functionplotter_digital.txt
22025: 070_pas_functionplotter_elliptic.txt
22026: 070_pas_function_helmholtz.txt
22027: 070_pas_textcheck_experimental.txt
22028: 071_pas_graphics.txt
22029: 071_pas_graphics_drawsym.txt
22030: 071_pas_graphics_drawsym_save.txt
22031: 071_pas_graphics_random.txt
22032: 072_pas_fractals.txt
22033: 072_pas_fractals_2.txt
22034: 072_pas_fractals_blackhole.txt
22035: 072_pas_fractals_perfomance.txt
22036: 072_pas_fractals_perfomance_new.txt
22037: 072_pas_fractals_perfomance_sharp.txt
22038: 072_pas_fractals_performance.txt
22039: 072_pas_fractals_performance_mX4.txt
22040: 073_pas_forms.txt
310_regex_decorator.TXT
312_ListView.txt
313_dmath_dll.txt
314_fundamentals4_tester.TXT
315_funcplot_dmath.TXT
316_cfileutils_cdatetime_tester.TXT
317_excel_export_tester.TXT
318_excel_export.TXT
318_excel_export2.TXT
318_excel_export3.TXT
318_excel_export3_tester.TXT
319_superfunctions_math.TXT
319_superfunctions_mathdefect.TXT
320_superfunctions.TXT
320_superfunctions2.TXT
321_SQL_Excel.txt
321_SQL_Excel2.txt
321_SQL_Excel_Export.txt
321_SQL_ExportExec.txt
321_SQL_ExportTest.txt
321_SQL_SAS_tester3.txt
321_SQL_SAS_tester3_selfcompile.txt
321_SQL_SAS_tester3_selfcompile2.txt
321_SQL_SAS_tester4.txt
321_SQL_SAS_updater.txt
322_timezones.TXT
323_datefind_fulltext_search.txt
323_datefind_fulltext_searchtester.txt
324_interfacenavi.TXT
325_ampelsteuerung.txt
325_analogclock.txt
326_world_analogclock.txt
326_world_analogclock2.txt
327_atomimage_clock.txt
328_starfield.txt
329_starfield2.txt
330_myclock.txt
330_myclock2.txt
331_SQL_DBfirebird4.txt
332_jprofiler.txt
332_jprofiler_form.txt
332_jprofiler_form2.txt
333_querybyexample.txt
333_querybyexample2.txt
334_jvutils_u.txt
335_atomimage5.txt
335_atomimage6.txt
335_atomimage7.txt
336_digiclock.txt
336_digiclock2.txt
336_digiclock2test.txt
336_digiclock3.txt
337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt
339_ntfs.txt
340_docutype.txt
340_logsimulation.txt
340_logsimulation2.txt
340_soundControltype.txt
341_blix_clock.txt
341_blix_clock2.txt
341_blix_clock_tester.txt
342_set_enumerator.txt
343_dice2.txt
344_pe_header.txt
344_pe_header2.txt
345_velocity.txt
346_conversions.txt
347_pictureview.txt
348_duallistview.txt
349_biginteger.txt
350_parserform.txt
351_chartform.txt
351_chartform2.txt
351_chartform3.txt
352_array_unittest.txt
353_smtp_email.txt
353_smtp_email2.txt
354_josephus.txt
355_life_of_PI.txt
356_3D_printer.txt
357_fplot.TXT
358_makesound.txt
359_charsetrules.TXT
360_allobjects.TXT
360_JvPaintFX.TXT
361_heartbeat_wave.TXT

22041: 074_pas_chartgenerator.txt
22042: 074_pas_chartgenerator_solution.txt
22043: 074_pas_chartgenerator_solution_back.txt
22044: 074_pas_charts.txt
22045: 075_bitmap_Artwork2.txt
22046: 075_pas_bitmappuzzle.txt
22047: 075_pas_bitmappuzzle24.prod.txt
22048: 075_pas_bitmappuzzle2_prod.txt
22049: 075_pas_bitmappuzzle3.txt
22050: 075_pas_bitmapsolve.txt
22051: 075_pas_bitmap_Artwork.txt
22052: 075_pas_puzzlepas_solution.txt
22053: 076_pas_3dcube.txt
22054: 076_pas_circle.txt
22055: 077_pas_mmshow.txt
22056: 078_pas_pi.txt
22057: 079_pas_3dcube_animation.txt
22058: 079_pas_3dcube_animation4.txt
22059: 079_pas_3dcube_plus.txt
22060: 080_pas_hanoi.txt
22061: 080_pas_hanoi2.txt
22062: 080_pas_hanoi2_file.txt
22063: 080_pas_hanoi2_sol.txt
22064: 080_pas_hanoi2_tester.txt
22065: 080_pas_hanoi2_tester_fast.txt
22066: 080_pas_hanoi3.txt
22067: 081_pas_chartist2.txt
22068: 082_pas_biorythmus.txt
22069: 082_pas_biorythmus_solution.txt
22070: 082_pas_biorythmus_solution_3.txt
22071: 082_pas_biorythmus_test.txt
22072: 083_pas_GITARRE.txt
22073: 083_pas_soundbox_tones.txt
22074: 084_pas_waves.txt
22075: 085_mxsinus_logo.txt
22076: 085_sinus_plot_waves.txt
22077: 086_pas_graph_arrow_heart.txt
22078: 087_bitmap_loader.txt
22079: 087_pas_bitmap_solution.txt
22080: 087_pas_bitmap_solution2.txt
22081: 087_pas_bitmap_subimage.txt
22082: 087_pas_bitmap_test.txt
22083: 088_pas_soundbox2_mp3.txt
22084: 088_pas_soundbox_mp3.txt
22085: 088_pas_sphere_2.txt
22086: 089_pas_gradient.txt
22087: 089_pas_maxland2.txt
22088: 090_pas_sudoku4.txt
22089: 090_pas_sudoku4_2.txt
22090: 091_pas_cube4.txt
22091: 092_pas_statistics4.txt
22092: 093_variance.txt
22093: 093_variance_debug.txt
22094: 094_pas_daysold.txt
22095: 094_pas_stat_date.txt
22096: 095_pas_ki_simulation.txt
22097: 096_pas_geisen_problem.txt
22098: 096_pas_montyhall_problem.txt
22099: 097_lotto_proofofconcept.txt
22100: 097_pas_lottocombinations_beat_plus.txt
22101: 097_pas_lottocombinations_beat_plus2.txt
22102: 097_pas_lottocombinations_universal.txt
22103: 097_pas_lottosimulation.txt
22104: 098_pas_chartgenerator_plus.txt
22105: 099_pas_3D_show.txt
22106: 200_big_numbers.txt
22107: 200_big_numbers2.txt
22108: 201_streamload_xml.txt
22109: 202_systemcheck.txt
22110: 203_webservice_simple_intftester.txt
22111: 204_webservice_simple.txt
22112: 205_future_value_service.txt
22113: 206_DTD_string_functions.txt
22114: 207_ibz2_async_process.txt
22115: 208_crc32_hash.txt
22116: 209_cryptohash.txt
22117: 210_public_private.txt
22118: 210_public_private_cryptosystem.txt
22119: 211_wipe_pattern.txt
22120: 211_wipe_pattern2.txt
22121: 211_wipe_pattern_solution.txt
22122: 212_pas_statisticmodule4.TXT
22123: 212_pas_statisticmoduletxt.TXT
22124: 212_statisticmodule4.txt
22125: 213_pas_BBP_Algo.txt
22126: 214_mxdocudemo.txt
22127: 214_mxdocudemo2.txt
22128: 214_mxdocudemo3.txt
22129: 215_hints_test.TXT

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

22130: 216_warnings_test.TXT
22131: 217_pas_heartbeat.txt
22132: 218_biorythmus_studio.txt
22133: 219_cipherbox.txt
22134: 219_crypt_source_comtest_solution.TXT
22135: 220_cipherbox_form.txt
22136: 220_cipherbox_form2.txt
22137: 221_bcd_explain.txt
22138: 222_memoform.txt
22139: 223_directorybox.txt
22140: 224_dialogs.txt
22141: 225_sprite_animation.txt
22142: 226_ASCII_Grid2.TXT
22143: 227_animation.txt
22144: 227_animation2.txt
22145: 228_android_calendar.txt
22146: 229_android_game.txt
22147: 229_android_game_tester.txt
22148: 230_DataProvider.txt
22149: 230_DataSetProvider.txt
22150: 230_DataSetXMLBackupScholz.txt
22151: 231_DBGrid_access.txt
22152: 231_DBGrid_XMLaccess.txt
22153: 231_DBGrid_XMLaccess2.txt
22154: 231_DBGrid_XMLaccess_locatetester.txt
22155: 231_DBGrid_XML_CDS_local.txt
22156: 232_outline.txt
22157: 232_outline_2.txt
22158: 233_modular_form.txt
22159: 234_debugoutform.txt
22160: 235_fastform.TXT
22161: 236_componentpower.txt
22162: 236_componentpower_back.txt
22163: 237_pas_4forms.txt
22164: 238_lottogen_form.txt
22165: 239_pas_sierpinski.txt
22166: 239_pas_sierpinski2.txt
22167: 240_unitGlobal_tester.txt
22168: 241_db3_sql_tutorial.txt
22169: 241_db3_sql_tutorial2.txt
22170: 241_db3_sql_tutorial2fix.txt
22171: 241_db3_sql_tutorial3.txt
22172: 241_db3_sql_tutorial3connect.txt
22173: 241_db3_sql_tutorial3_fptest.txt
22174: 241_RTL_SET2.txt
22175: 241_RTL_SET2_tester.txt
22176: 242_Component_Control.txt
22177: 243_tutorial_loader.txt
22178: 244_script_loader_loop.txt
22179: 245_formapp2.txt
22180: 245_formapp2_tester.txt
22181: 245_formapp2_testerX.txt
22182: 246_httpapp.txt
22183: 247_datecalendar.txt
22184: 248_ASCII_Grid2_sorted.TXT
22185: 249_picture_grid.TXT
22186: 250_tipsandtricks2.txt
22187: 250_tipsandtricks3.txt
22188: 250_tipsandtricks3api.txt
22189: 250_tipsandtricks3_admin_elevation.txt
22190: 250_tipsandtricks3_tester.txt
22191: 250_tipsandtricks4_tester.txt
22192: 250_tipsandtricks4_tester2.txt
22193: 251_compare_noise_gauss.txt
22194: 251_whitenoise.txt
22195: 251_whitenoise2.txt
22196: 252_hilbert_turtle.txt
22197: 252_pas_hilbert.txt
22198: 253_opearatingsystem3.txt
22199: 254_dynarrays.txt
22200: 255_einstein.txt
22201: 256_findconsts_of_EXE.txt
22202: 256_findfunctions2_of_EXE.txt
22203: 256_findfunctions2_of_EXEaverp.txt
22204: 256_findfunctions2_of_EXEspec.txt
22205: 256_findfunctions3.txt
22206: 256_findfunctions_of_EXE.txt
22207: 257_AES_Cipher.txt
22208: 258_AES_cryptobox.txt
22209: 258_AES_cryptobox2.txt
22210: 258_AES_cryptobox2_passdlg.txt
22211: 259_AES_crypt_directory.txt
22212: 260_sendmessage_2.TXT
22213: 260_sendmessage_beta.TXT
22214: 261_probability.txt
22215: 262_mxoutputdemo4.txt
22216: 263_async_sound.txt
22217: 264_vclutils.txt
22218: 264_VCL_utils2.txt

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

```
22219: 265_timer_API.txt                                  458_atomimageX.txt
22220: 266_serial_interface.txt                           459_cindyfunc.txt
22221: 266_serial_interface2.txt                          459_cindyfunc2.txt
22222: 266_serial_interface3.txt                          460_TopTenFunctions.txt
22223: 267_ackermann_rec.txt                              461_sqlform_calwin.txt
22224: 267_ackermann_variants.txt                         462_caesarcipher.txt
22225: 268_DBGrid_tree.txt                                463_global_exception.txt
22226: 269_record_grid.TXT                                464_function_procedure.txt
22227: 270_Jedi_FunctionPower.txt                         464_function_procedure2.txt
22228: 270_Jedi_FunctionPowertester.txt                   464_function_procedure3.txt
22229: 271_closures_study.txt                             465_U_HexView.txt
22230: 271_closures_study_workingset2.txt                 466_moon.txt
22231: 272_pas_function_show.txt                          466_moon_inputquery.txt
22232: 273_pas_function_show2.txt                         4671_cardmagic.txt
22233: 274_library_functions.txt                          467_helmholtz_graphic.txt
22234: 275_turtle_language.txt                            468_URLMon.txt
22235: 275_turtle_language_save.txt                       468_URLMon2.txt
22236: 276_save_algo.txt                                  469_formarrow.txt
22237: 276_save_algo2.txt                                 469_formarrow_datepicker.txt
22238: 277_functionsfor39.txt                             469_formarrow_datepicker_ibz_result.txt
22239: 278_DB_Dialogs.TXT                                 469_ibzresult.txt
22240: 279_hexer2.TXT                                     470_DFFUtils_compiled.txt
22241: 279_hexer2macro.TXT                                470_DFFUtils_ScrollingLED.txt
22242: 279_hexer2macroback.TXT                            470_Oscilloscope.txt
22243: 280_UML_process.txt                                470_Oscilloscope_code.txt
22244: 280_UML_process_knabe2.txt                         471_cardmagic.txt
22245: 280_UML_process_knabe3.txt                         471_cardmagic2.txt
22246: 280_UML_process_TIM_Botzenhardt.txt                472_allcards.TXT
22247: 280_UML_TIM_Seitz.txt                              473_comboset.txt
22248: 281_picturepuzzle.txt                              474_wakeonlan.txt
22249: 281_picturepuzzle2.txt                             474_wakeonlan2.txt
22250: 281_picturepuzzle3.txt                             476_getscripttest.txt
22251: 281_picturepuzzle4.txt                             477_filenameonly.txt
22252: 479_inputquery.txt                                 480_regex_pathfinder.txt
22253: 480_regex_pathfinder2.txt                          481_processList.txt
22254: 482 processPipe.txt                                482 processPipeGCC.txt
22255: 483_PathFuncTest_mX.txt                            484_filefinder3.txt
22256: 485_InnoFunc.txt                                   486_VideoGrabber.txt
22257: 487_asyncKeyState.txt                              488_asyncTerminal.txt
22258: 489_simpleComport.txt                              490_webCamproc.txt
22259: 491_analogmeter.txt                                492_snowflake.txt
22260:
22261:
22262: Web Script Examples:
22263:
22264: http://www.softwareschule.ch/examples/performer.txt';
22265: http://www.softwareschule.ch/examples/turtle.txt';
22266: http://www.softwareschule.ch/examples/SQLExport.txt';
22267: http://www.softwareschule.ch/examples/Richter.txt';
22268: http://www.softwareschule.ch/examples/checker.txt';
22269: http://www.softwareschule.ch/examples/demoscript.txt';
22270: http://www.softwareschule.ch/examples/ibzresult.txt';
22271: http://www.softwareschule.ch/examples/performindex.txt
22272: http://www.softwareschule.ch/examples/processlist.txt
22273: http://www.softwareschule.ch/examples/game.txt
22274:
22275: SHA1:  maXbox3.exe 4C7A84045994D69DA0131D3E518D9B4901F20C5D
22276: CRC32: maXbox3.exe EF0509C
22277: Ref:
22278:     1. writeln(SHA1(Exepath+'\maxbox3.exe'))
22279:     2. shdig: TSHA1Digest;
22280:        shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
22281:        for i:= 0 to 19 do write(BytetoHex(shdig[i]));
22282:
22283:     3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
22284:
22285:   ---- bigbitbox code_cleared_checked----
```