```
 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: 22866944 V3.9.9.98 September 2014 To EKON/BASTA/JAX/IBZ/SWS/EU/DD
10: ***************Now the Funclist*****************************************
11: Funclist Function : 13271 //10766 //10165 (7648)
12: ***************Now the Proclist*****************************************
13: Proclist Procedure Size is: 8274  //6792 //6401 4752
14: ***************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1338 //995 //
16: def head:max: maXbox7: 02.09.2014 09:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: --------------------------------------------------------------------------------
20: Funclist total Size all is: 22883! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 23055
22: ASize of EXE: 22866944 (22041600) (16586240) (13511680)
23: SHA1 Hash of maXbox 3.9.9.98: D0EC95326FE1ABD9D441F137336D00CF4BC77CAB
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 CompareValueE(const A, B: Extended; Epsilon: Extended = 0): TValueRelationship; overload;
406: function CompareValueD(const A, B: Double; Epsilon: Double = 0): TValueRelationship; overload;
407: function CompareValueS(const A, B: Single; Epsilon: Single = 0): TValueRelationship; overload;
408: function CompareValueI(const A, B: Integer): TValueRelationship; overload;
409: function CompareValueI64(const A, B: Int64): TValueRelationship; overload;
410: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean
411: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean
412: Function ComponentTypeToString( const ComponentType : DWORD) : string
413: Function ComposeDateTime(Date,Time : TDateTime) : TDateTime;');
414: Function CompToCurrency( Value : Comp) : Currency
415: Function CompToDouble( Value : Comp) : Double
416: function ComputeFileCRC32(const FileName : String) : Integer;
417: function ComputeSHA256(astr: string; amode: char): string)  //mode F:File, S:String
418: function ComputeSHA512(astr: string; amode: char): string)  //mode F:File, S:String
419: function Concat(s: string): string
420: Function ConnectAndGetAll : string
421: Function Connected : Boolean
422: function constrain(x, a, b: integer): integer;
423: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult
424: Function ConstraintsDisabled : Boolean
425: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
426: Function ContainsState( oState : TniRegularExpressionState) : boolean
427: Function ContainsStr( const AText, ASubText : string) : Boolean
428: Function ContainsText( const AText, ASubText : string) : Boolean
429: Function ContainsTransition( oTransition : TniRegularExpressionTransition) : boolean
430: Function Content : string
431: Function ContentFromStream( Stream : TStream) : string
432: Function ContentFromString( const S : string) : string
433: Function CONTROLSDISABLED : BOOLEAN
434: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
435: Function Convert1( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
436: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
437: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
438: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
439: Function ConvertWriteStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
440: Function ConvFamilyToDescription( const AFamily : TConvFamily) : string
441: Function ConvTypeToDescription( const AType : TConvType) : string
442: Function ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
443: Function ConvTypeToFamily( const AType : TConvType) : TConvFamily;
444: Function ConvAdd(const AVal:Dbl;const AType1:TConvType;const AVal2:Dbl;const AType2,
     AResultType:TConvType): Double
```

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

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

```
612: Function DegToGrad( const Degrees : Extended) : Extended
613: Function DegToGrad( const Value : Extended) : Extended;
614: Function DegToGrad1( const Value : Double) : Double;
615: Function DegToGrad2( const Value : Single) : Single;
616: Function DegToRad( const Degrees : Extended) : Extended
617: Function DegToRad( const Value : Extended) : Extended;
618: Function DegToRad1( const Value : Double) : Double;
619: Function DegToRad2( const Value : Single) : Single;
620: Function DelChar( const pStr : string; const pChar : Char) : string
621: Function DelEnvironmentVar( const Name : string) : Boolean
622: Function Delete( const MsgNum : Integer) : Boolean
623: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
624: Function DeleteFile(const FileName: string): boolean)
625: Function DeleteFileEx( FileName : string; Flags : FILEOP_FLAGS) : Boolean
626: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
627: Function DelimiterPosn1(const sString:string;const sDelimiters:string;out cDelimiter: char) : integer;
628: Function DelSpace( const pStr : string) : string
629: Function DelString( const pStr, pDelStr : string) : string
630: Function DelTree( const Path : string) : Boolean
631: Function Depth : Integer
632: Function Description : string
633: Function DescriptionsAvailable : Boolean
634: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily) : Boolean
635: Function DescriptionToConvType( const ADescription : string; out AType : TConvType) : Boolean;
636: Function DescriptionToConvType1(const AFamil:TConvFamily;const ADescr:string;out AType:TConvType):Boolean;
637: Function DetectUTF8Encoding( const s : UTF8String) : TEncodeType
638: Function DialogsToPixelsX( const Dialogs : Word) : Word
639: Function DialogsToPixelsY( const Dialogs : Word) : Word
640: Function Digits( const X : Cardinal) : Integer
641: Function DirectoryExists( const Name : string) : Boolean
642: Function DirectoryExists( Directory : string) : Boolean
643: Function DiskFree( Drive : Byte) : Int64
644: function DiskFree(Drive: Byte): Int64)
645: Function DiskInDrive( Drive : Char) : Boolean
646: Function DiskSize( Drive : Byte) : Int64
647: function DiskSize(Drive: Byte): Int64)
648: Function DISPATCHCOMMAND( ACOMMAND : WORD) : BOOLEAN
649: Function DispatchEnabled : Boolean
650: Function DispatchMask : TMask
651: Function DispatchMethodType : TMethodType
652: Function DISPATCHPOPUP( AHANDLE : HMENU) : BOOLEAN
653: Function DispatchRequest( Sender : TObject; Request : TWebRequest; Response : TWebResponse) : Boolean
654: Function DisplayCase( const S : String) : String
655: Function DisplayRect( Code : TDisplayCode) : TRect
656: Function DisplayRect( TextOnly : Boolean) : TRect
657: Function DisplayStream( Stream : TStream) : string
658: TBufferCoord', 'record Char : integer; Line : integer; end
659: TDisplayCoord', 'record Column : integer; Row : integer; end
660: Function DisplayCoord( AColumn, ARow : Integer) : TDisplayCoord
661: Function BufferCoord( AChar, ALine : Integer) : TBufferCoord
662: Function DomainName( const AHost : String) : String
663: Function DosPathToUnixPath( const Path : string) : string
664: Function DottedLineTo( const Canvas : TCanvas; const X, Y : Integer) : Boolean;
665: Function DoubleDecliningBalance( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
666: Function DoubleToBcd( const AValue : Double) : TBcd;
667: Function DoubleToHex( const D : Double) : string
668: Function DoUpdates : Boolean
669: Function Dragging: Boolean;
670: Function DrawCaption( p1 : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL
671: Function DrawAnimatedRects( hwnd : HWND; idAni : Integer; const lprcFrom, lprcTo : TRect) : BOOL
672: Function DrawEdge( hdc : HDC; var qrc : TRect; edge : UINT; grfFlags : UINT) : BOOL
673: Function DrawFrameControl( DC : HDC; const Rect : TRect; uType, uState : UINT) : BOOL
674: {Works like InputQuery but displays 2edits. If PasswordChar <> #0, the second edit's PasswordChar is set}
675: Function DualInputQuery(const ACapt,Prpt1,Prpt2:string;var AVal1,AVal2:string;PasswrdChar:Char= #0):Bool;
676: Function DupeString( const AText : string; ACount : Integer) : string
677: Function Edit : Boolean
678: Function EditCaption : Boolean
679: Function EditText : Boolean
680: Function EditFolderList( Folders : TStrings) : Boolean
681: Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
682: Function Elapsed( const Update : Boolean) : Cardinal
683: Function EnableProcessPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
684: Function EnableThreadPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
685: Function EncodeDate( Year, Month, Day : Word) : TDateTime
686: function EncodeDate(Year, Month, Day: Word): TDateTime;
687: Function EncodeDateDay( const AYear, ADayOfYear : Word) : TDateTime
688: Function EncodeDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : TDateTime
689: Function EncodeDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSecond: Word): TDateTime
690: Function EncodeDateWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
691: Function EncodeDayOfWeekInMonth( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word) : TDateTime
692: Function EncodeString( s : string) : string
693: Function DecodeString( s : string) : string
694: Function EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
695: function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
696: Function EndIP : String
697: Function EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
698: Function EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
699: Function EndOfAMonth( const AYear, AMonth : Word) : TDateTime
700: Function EndOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
```

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

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

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

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

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

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

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

```
1311: Function GetShellVersion : Cardinal
1312: function getShortDayNames: string)
1313: Function GetShortHint(const hint: string): string
1314: function getShortMonthNames: string)
1315: Function GetSizeOfFile( const FileName : string) : Int64;
1316: Function GetSizeOfFile1( Handle : THandle) : Int64;
1317: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1318: Function GetStartmenuFolder : string
1319: Function GetStartupFolder : string
1320: Function GetStringProperty( Instance : TPersistent; const PropName : string) : WideString
1321: Function GetSuccessor( cChar : char) : TniRegularExpressionState
1322: Function GetSwapFileSize : Integer
1323: Function GetSwapFileUsage : Integer
1324: Function GetSystemLocale : TIdCharSet
1325: Function GetSystemMetrics( nIndex : Integer) : Integer
1326: Function GetSystemPathSH(Folder: Integer): TFilename ;
1327: Function GetTableNameFromQuery( const SQL : Widestring) : Widestring
1328: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1329: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1330: Function GetTasksList( const List : TStrings) : Boolean
1331: Function getTeamViewerID: string;
1332: Function GetTemplatesFolder : string
1333: Function GetText : PwideChar
1334: function GetText:PChar
1335: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1336: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1337: Function GetTextItem( const Value : string) : Longint
1338: function GETTEXTLEN:INTEGER
1339: Function GetThreadLocale: Longint; stdcall
1340: Function GetCurrentThreadID: LongWord; stdcall;
1341: Function GetTickCount : Cardinal
1342: Function GetTickDiff( const AOldTickCount, ANewTickCount : Cardinal) : Cardinal
1343: Function GetTicketNr : longint
1344: Function GetTime : Cardinal
1345: Function GetTime : TDateTime
1346: Function GetTimeout : Integer
1347: Function GetTimeStr: String
1348: Function GetTimeString: String
1349: Function GetTodayFiles(startdir, amask: string): TStringlist;
1350: Function getTokenCounts : integer
1351: Function GetTotalPageFileMemory : Integer
1352: Function GetTotalPhysicalMemory : Integer
1353: Function GetTotalVirtualMemory : Integer
1354: Function GetUniqueFileName( const APath, APrefix, AExt : String) : String
1355: Function GetUseNowForDate : Boolean
1356: Function GetUserDomainName( const CurUser : string) : string
1357: Function GetUserName : string
1358: Function GetUserName: string;
1359: Function GetUserObjectName( hUserObject : THandle) : string
1360: Function GetValueBitmap( Value : TBitmap) : TBitmap
1361: Function GetValueMSec : Cardinal
1362: Function GetValueStr : String
1363: Function GetVersion: int;
1364: Function GetVersionString(FileName: string): string;
1365: Function getVideoDrivers: string;
1366: Function GetVisibleNode( Index : LongInt) : TOutlineNode
1367: Function GetVolumeFileSystem( const Drive : string) : string
1368: Function GetVolumeName( const Drive : string) : string
1369: Function GetVolumeSerialNumber( const Drive : string) : string
1370: Function GetWebAppServices : IWebAppServices
1371: Function GetWebRequestHandler : IWebRequestHandler
1372: Function GetWindowCaption( Wnd : HWND) : string
1373: Function GetWindowDC(hdwnd: HWND): HDC;
1374: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1375: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1376: Function GetWindowsComputerID : string
1377: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1378: Function GetWindowsFolder : string
1379: Function GetWindowsServicePackVersion : Integer
1380: Function GetWindowsServicePackVersionString : string
1381: Function GetWindowsSystemFolder : string
1382: Function GetWindowsTempFolder : string
1383: Function GetWindowsUserID : string
1384: Function GetWindowsVersion : TWindowsVersion
1385: Function GetWindowsVersionString : string
1386: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1387: Function GMTToLocalDateTime( S : string) : TDateTime
1388: Function GotoKey : Boolean
1389: Function GradToCycle( const Grads : Extended) : Extended
1390: Function GradToDeg( const Grads : Extended) : Extended
1391: Function GradToDeg( const Value : Extended) : Extended;
1392: Function GradToDeg1( const Value : Double) : Double;
1393: Function GradToDeg2( const Value : Single) : Single;
1394: Function GradToRad( const Grads : Extended) : Extended
1395: Function GradToRad( const Value : Extended) : Extended;
1396: Function GradToRad1( const Value : Double) : Double;
1397: Function GradToRad2( const Value : Single) : Single;
1398: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1399: Function GreenComponent( const Color32 : TColor32) : Integer
```

```
1400: function GUIDToString(const GUID: TGUID): string)
1401: Function HandleAllocated : Boolean
1402: function HandleAllocated: Boolean;
1403: Function HandleRequest : Boolean
1404: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1405: Function HarmonicMean( const X : TDynFloatArray) : Float
1406: Function HasAsParent( Value : TTreeNode) : Boolean
1407: Function HASCHILDDEFS : BOOLEAN
1408: Function HasCurValues : Boolean
1409: Function HasExtendCharacter( const s : UTF8String) : Boolean
1410: Function HasFormat( Format : Word) : Boolean
1411: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1412: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1413: Function HashValue(AStream: TStream): LongWord
1414: Function HashValue(AStream: TStream): Word
1415: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1416: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1417: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1418: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1419: Function HashValue16( const ASrc : string) : Word;
1420: Function HashValue16stream( AStream : TStream) : Word;
1421: Function HashValue32( const ASrc : string) : LongWord;
1422: Function HashValue32Stream( AStream : TStream) : LongWord;
1423: Function HasMergeConflicts : Boolean
1424: Function hasMoreTokens : boolean
1425: Function HASPARENT : BOOLEAN
1426: function HasParent: Boolean
1427: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1428: Function HasUTF8BOM( S : TStream) : boolean;
1429: Function HasUTF8BOM1( S : AnsiString) : boolean;
1430: Function Haversine( X : Float) : Float
1431: Function Head( s : string; const subs : string; var tail : string) : string
1432: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1433: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1434: function HELPJUMP(JUMPID:STRING):BOOLEAN
1435: Function HeronianMean( const a, b : Float) : Float
1436: function HexStrToStr(Value: string): string;
1437: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1438: function HexToBin2(HexNum: string): string;
1439: Function HexToDouble( const Hex : string) : Double
1440: function HexToInt(hexnum: string): LongInt;
1441: function HexToStr(Value: string): string;
1442: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1443: function Hi(vdat: word): byte;
1444: function HiByte(W: Word): Byte)
1445: function High: Int64;
1446: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1447: function HINSTANCE: longword;
1448: function HiWord(l: DWORD): Word)
1449: function HMODULE: longword;
1450: Function HourOf( const AValue : TDateTime) : Word
1451: Function HourOfTheDay( const AValue : TDateTime) : Word
1452: Function HourOfTheMonth( const AValue : TDateTime) : Word
1453: Function HourOfTheWeek( const AValue : TDateTime) : Word
1454: Function HourOfTheYear( const AValue : TDateTime) : Word
1455: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1456: Function HourSpan( const ANow, AThen : TDateTime) : Double
1457: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1458: Function HTMLDecode( const AStr : String) : String
1459: Function HTMLEncode( const AStr : String) : String
1460: Function HTMLEscape( const Str : string) : string
1461: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1462: Function HTTPDecode( const AStr : String) : string
1463: Function HTTPEncode( const AStr : String) : string
1464: Function Hypot( const X, Y : Extended) : Extended
1465: Function IBMax( n1, n2 : Integer) : Integer
1466: Function IBMin( n1, n2 : Integer) : Integer
1467: Function IBRandomString( iLength : Integer) : String
1468: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1469: Function IBStripString( st : String; CharsToStrip : String) : String
1470: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1471: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1472: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1473: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1474: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1475: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1476:  Function RandomString( iLength : Integer) : String');
1477:  Function RandomInteger( iLow, iHigh : Integer) : Integer');
1478:  Function StripString( st : String; CharsToStrip : String) : String');
1479:  Function FormatIdentifier( Dialect : Integer; Value : String) : String');
1480:  Function FormatIdentifierValue( Dialect : Integer; Value : String) : String');
1481:  Function ExtractIdentifier( Dialect : Integer; Value : String) : String');
1482:  Function QuoteIdentifier( Dialect : Integer; Value : String) : String');
1483:  Function AddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string;
1484:  Procedure DecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String;
1485:  function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;
1486: Function IconToBitmap( Ico : HICON) : TBitmap
1487: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1488: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
```

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

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

```
1665: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1666: Function IsMatch( const Pattern, Text : string) : Boolean  //Grep like RegEx
1667: Function IsMainAppWindow( Wnd : HWND) : Boolean
1668: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1669: function IsMemoryManagerSet: Boolean)
1670: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1671: function IsMultiThread: Boolean)
1672: Function IsNumeric( AChar : Char) : Boolean;
1673: Function IsNumeric2( const AString : string) : Boolean;
1674: Function IsNTFS: Boolean;
1675: Function IsOctal( AChar : Char) : Boolean;
1676: Function IsOctalString(const AString: string) : Boolean;
1677: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1678: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1679: Function IsPM( const AValue : TDateTime) : Boolean
1680: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1681: Function IsPortAvailable( ComNum : Cardinal) : Boolean');
1682: Function IsCOMPortReal( ComNum : Cardinal) : Boolean');
1683: Function IsCOM( ComNum : Cardinal) : Boolean');
1684: Function IsCOMPort: Boolean');
1685: Function IsPrimeFactor( const F, N : Cardinal) : Boolean
1686: Function IsPrimeRM( N : Cardinal) : Boolean  //rabin miller
1687: Function IsPrimeTD( N : Cardinal) : Boolean  //trial division
1688: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1689: Function ISqrt( const I : Smallint) : Smallint
1690: Function IsReadOnly(const Filename: string): boolean;
1691: Function IsRectEmpty( const Rect : TRect) : Boolean
1692: Function IsRectEmpty(const Rect: TRect): Boolean)
1693: Function IsRelativePrime( const X, Y : Cardinal) : Boolean
1694: Function ISRIGHTTOLEFT : BOOLEAN
1695: function IsRightToLeft: Boolean
1696: Function IsSameDay( const AValue, ABasis : TDateTime) : Boolean
1697: Function ISSEQUENCED : BOOLEAN
1698: Function IsSystemModule( const Module : HMODULE) : Boolean
1699: Function IsSystemResourcesMeterPresent : Boolean
1700: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1701: Function IsToday( const AValue : TDateTime) : Boolean
1702: function IsToday(const AValue: TDateTime): Boolean;
1703: Function IsTopDomain( const AStr : string) : Boolean
1704: Function IsUTF8LeadByte( Lead : Char) : Boolean
1705: Function IsUTF8String( const s : UTF8String) : Boolean
1706: Function IsUTF8TrailByte( Lead : Char) : Boolean
1707: Function ISVALIDCHAR( INPUTCHAR : CHAR) : BOOLEAN
1708: Function IsValidDate( const AYear, AMonth, ADay : Word) : Boolean
1709: Function IsValidDateDay( const AYear, ADayOfYear : Word) : Boolean
1710: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : Boolean
1711: Function IsValidDateTime(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1712: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) : Boolean
1713: Function IsValidIdent( Ident : string) : Boolean
1714: function IsValidIdent1(const Ident: string; AllowDots: Boolean): Boolean)
1715: Function IsValidIP( const S : String) : Boolean
1716: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) : Boolean
1717: Function IsValidISBN( const ISBN : AnsiString) : Boolean
1718: Function IsVariantManagerSet: Boolean; //deprecated;
1719: Function IsVirtualPcGuest : Boolean;
1720: Function IsVmWareGuest : Boolean;
1721: Function IsVCLControl(Handle: HWnd): Boolean;
1722: Function IsWhiteString( const AStr : String) : Boolean
1723: Function IsWindowResponding( Wnd : HWND; Timeout : Integer) : Boolean
1724: Function IsWoW64: boolean;
1725: Function IsWin64: boolean;
1726: Function IsWow64String(var s: string): Boolean;
1727: Function IsWin64String(var s: string): Boolean;
1728: Function IsWindowsVista: boolean;
1729: Function isPowerof2(num: int64): boolean;
1730: Function powerOf2(exponent: integer): int64;
1731: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1732: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1733: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1734: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1735: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1736: Function ItemRect( Index : Integer) : TRect
1737: function ITEMRECT(INDEX:INTEGER):TRECT
1738: Function ItemWidth( Index : Integer) : Integer
1739: Function JavahashCode(val: string): Integer;
1740: Function JosephusG(n,k: integer; var graphout: string): integer;
1741: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1742: Function JustName(PathName : string) : string;  //in path and ext
1743: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1744: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1745: Function KeepAlive : Boolean
1746: Function KeysToShiftState(Keys: Word): TShiftState;
1747: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1748: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1749: Function KeyboardStateToShiftState: TShiftState; overload;
1750: Function Languages : TLanguages
1751: Function Last : TClass
1752: Function Last : TComponent
1753: Function Last : TObject
```

```
1754: Function LastDelimiter( Delimiters, S : string) : Integer
1755: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1756: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1757: Function Latitude2WGS84(lat: double): double;
1758: Function LCM(m,n:longint):longint;
1759: Function LCMJ( const X, Y : Cardinal) : Cardinal
1760: Function Ldexp( const X : Extended; const P : Integer) : Extended
1761: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1762: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1763: function Length: Integer;
1764: Procedure LetFileList(FileList: TStringlist; apath: string);
1765: function lengthmp3(mp3path: string):integer;
1766: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1767: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1768: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1769: function LineStart(Buffer, BufPos: PChar): PChar
1770: function LineStart(Buffer, BufPos: PChar): PChar)
1771: function ListSeparator: char;
1772: function Ln(x: Extended): Extended;
1773: Function LnXP1( const X : Extended) : Extended
1774: function Lo(vdat: word): byte;
1775: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1776: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1777: Function LoadFileAsString( const FileName : string) : string
1778: Function LoadFromFile( const FileName : string) : TBitmapLoader
1779: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1780: Function LoadPackage(const Name: string): HMODULE
1781: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1782: Function LoadStr( Ident : Integer) : string
1783: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1784: Function LoadWideStr( Ident : Integer) : WideString
1785: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1786: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1787: Function LockServer( fLock : LongBool) : HResult
1788: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1789: Function Log( const X : Extended) : Extended
1790: Function Log10( const X : Extended) : Extended
1791: Function Log2( const X : Extended) : Extended
1792: function LogBase10(X: Float): Float;
1793: Function LogBase2(X: Float): Float;
1794: Function LogBaseN(Base, X: Float): Float;
1795: Function LogN( const Base, X : Extended) : Extended
1796: Function LogOffOS : Boolean
1797: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1798: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1799: Function LongDateFormat: string;
1800: function LongTimeFormat: string;
1801: Function LongWordToFourChar( ACardinal : LongWord) : string
1802: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1803: Function LookupName( const name : string) : TInAddr
1804: Function LookupService( const service : string) : Integer
1805: function Low: Int64;
1806: Function LowerCase( S : string) : string
1807: Function Lowercase(s : AnyString) : AnyString;
1808: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1809: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1810: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1811: function MainInstance: longword
1812: function MainThreadID: longword
1813: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1814: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1815: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1816: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1817: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1818: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1819: Function MakeFile(const FileName: string): integer)');
1820: function MakeLong(A, B: Word): Longint)
1821: Function MakeTempFilename( const APath : String) : string
1822: Function MakeValidFileName( const Str : string) : string
1823: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1824: function MakeWord(A, B: Byte): Word)
1825: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1826: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1827: Function MapValues( Mapping : string; Value : string) : string
1828: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1829: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1830: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1831: Function MaskGetFldSeparator( const EditMask : string) : Integer
1832: Function MaskGetMaskBlank( const EditMask : string) : Char
1833: Function MaskGetMaskSave( const EditMask : string) : Boolean
1834: Function MaskIntlLiteralToChar( IChar : Char) : Char
1835: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1836: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1837: Function MaskString( Mask, Value : String) : String
1838: Function Match( const sString : string) : TniRegularExpressionMatchResul
1839: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1840: Function Matches( const Filename : string) : Boolean
1841: Function MatchesMask( const Filename, Mask : string) : Boolean
```

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

```
1926: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1927: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1928: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1929: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1930: Function MinValue( const Data : array of Double) : Double
1931: function MinValue(const Data: array of Double): Double)
1932: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1933: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1934: Function ModFloat( const X, Y : Float) : Float
1935: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1936: Function Modify( const Key : string; Value : Integer) : Boolean
1937: Function ModuleCacheID : Cardinal
1938: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1939: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1940: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1941: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1942: Function MonthOf( const AValue : TDateTime) : Word
1943: Function MonthOfTheYear( const AValue : TDateTime) : Word
1944: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1945: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1946: Function MonthStr( DateTime : TDateTime) : string
1947: Function MouseCoord( X, Y : Integer) : TGridCoord
1948: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1949: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1950: Function MoveNext : Boolean
1951: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1952: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1953: Function Name : string
1954: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
      Double;PaymentTime:TPaymentTime):Extended
1955: function NetworkVolume(DriveChar: Char): string
1956: Function NEWBOTTOMLINE : INTEGER
1957: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1958: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
      TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1959: Function NEWLINE : TMENUITEM
1960: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1961: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
      Right:PExprNode):PExprNode
1962: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
      const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1963: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
1964: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
      TMenuItem;AENABLED:BOOL): TMENUITEM
1965: Function NEWTOPLINE : INTEGER
1966: Function Next : TIdAuthWhatsNext
1967: Function NextCharIndex( S : String; Index : Integer) : Integer
1968: Function NextRecordSet : TCustomSQLDataSet
1969: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1970: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
1971: Function NextToken : Char
1972: Function nextToken : WideString
1973: function NextToken:Char
1974: Function Norm( const Data : array of Double) : Extended
1975: Function NormalizeAngle( const Angle : Extended) : Extended
1976: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
1977: Function NormalizeRect( const Rect : TRect) : TRect
1978: function NormalizeRect(const Rect: TRect): TRect;
1979: Function Now : TDateTime
1980: function Now2: tDateTime
1981: Function NumProcessThreads : integer
1982: Function NumThreadCount : integer
1983: Function NthDayOfWeek( const AValue : TDateTime) : Word
1984: Function NtProductType : TNtProductType
1985: Function NtProductTypeString : string
1986: function Null: Variant;
1987: Function NullPoint : TPoint
1988: Function NullRect : TRect
1989: Function Null2Blank(aString:String):String;
1990: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
1991: Function NumIP : integer
1992: function Odd(x: Longint): boolean;
1993: Function OffsetFromUTC : TDateTime
1994: Function OffsetPoint( const P, Offset : TPoint) : TPoint
1995: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
1996: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
1997: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
1998: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
1999: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
2000: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
2001: function OpenBit:Integer
2002: Function OpenDatabase : TDatabase
2003: Function OpenDatabase( const DatabaseName : string) : TDatabase
2004: Procedure OpenDir(adir: string);
2005: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
2006: Function OpenObject( Value : PChar) : Boolean;
2007: Function OpenObject1( Value : string) : Boolean;
2008: Function OpenSession( const SessionName : string) : TSession
```

```
2009: Function OpenVolume( const Drive : Char) : THandle
2010: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
2011: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
2012: Function OrdToBinary( const Value : Byte) : string;
2013: Function OrdToBinary1( const Value : Shortint) : string;
2014: Function OrdToBinary2( const Value : Smallint) : string;
2015: Function OrdToBinary3( const Value : Word) : string;
2016: Function OrdToBinary4( const Value : Integer) : string;
2017: Function OrdToBinary5( const Value : Cardinal) : string;
2018: Function OrdToBinary6( const Value : Int64) : string;
2019: Function OSCheck( RetVal : Boolean) : Boolean
2020: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
2021: Function OSIdentToString( const OSIdent : DWORD) : string
2022: Function Output: Text)
2023: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
2024: Function Owner : TCustomListView
2025: function Owner : TPersistent
2026: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
2027: Function PadL( pStr : String; pLth : integer) : String
2028: Function Padl(s : AnyString;I : longInt) : AnyString;
2029: Function PadLCh( pStr : String; pLth : integer; pChr : char) : String
2030: Function PadR( pStr : String; pLth : integer) : String
2031: Function Padr(s : AnyString;I : longInt) : AnyString;
2032: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
2033: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2034: Function Padz(s : AnyString;I : longInt) : AnyString;
2035: Function PaethPredictor( a, b, c : Byte) : Byte
2036: Function PARAMBYNAME( const VALUE : String) : TPARAM
2037: Function ParamByName( const Value : WideString) : TParameter
2038: Function ParamCount: Integer
2039: Function ParamsEncode( const ASrc : string) : string
2040: function ParamStr(Index: Integer): string)
2041: Function ParseDate( const DateStr : string) : TDateTime
2042: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
2043: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2044: Function PathAddExtension( const Path, Extension : string) : string
2045: Function PathAddSeparator( const Path : string) : string
2046: Function PathAppend( const Path, Append : string) : string
2047: Function PathBuildRoot( const Drive : Byte) : string
2048: Function PathCanonicalize( const Path : string) : string
2049: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2050: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2051: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2052: Function PathEncode( const ASrc : string) : string
2053: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2054: Function PathExtractFileNameNoExt( const Path : string) : string
2055: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2056: Function PathGetDepth( const Path : string) : Integer
2057: Function PathGetLongName( const Path : string) : string
2058: Function PathGetLongName2( Path : string) : string
2059: Function PathGetShortName( const Path : string) : string
2060: Function PathIsAbsolute( const Path : string) : Boolean
2061: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2062: Function PathIsDiskDevice( const Path : string) : Boolean
2063: Function PathIsUNC( const Path : string) : Boolean
2064: Function PathRemoveExtension( const Path : string) : string
2065: Function PathRemoveSeparator( const Path : string) : string
2066: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2067: Function Peek : Pointer
2068: Function Peek : TObject
2069: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2070: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2071: function Permutation(npr, k: integer): extended;
2072: function PermutationInt(npr, k: integer): Int64;
2073: Function PermutationJ( N, R : Cardinal) : Float
2074: Function Pi : Extended;
2075: Function PiE : Extended;
2076: Function PixelsToDialogsX( const Pixels : Word) : Word
2077: Function PixelsToDialogsY( const Pixels : Word) : Word
2078: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2079: Function Point( X, Y : Integer) : TPoint
2080: function Point(X, Y: Integer): TPoint)
2081: Function PointAssign( const X, Y : Integer) : TPoint
2082: Function PointDist( const P1, P2 : TPoint) : Double;
2083: function PointDist(const P1,P2: TFloatPoint): Double;
2084: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2085: function PointDist2(const P1,P2: TPoint): Double;
2086: Function PointEqual( const P1, P2 : TPoint) : Boolean
2087: Function PointIsNull( const P : TPoint) : Boolean
2088: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2089: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2090: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2091: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2092: Function Pop : Pointer
2093: Function Pop : TObject
2094: Function PopnStdDev( const Data : array of Double) : Extended
2095: Function PopnVariance( const Data : array of Double) : Extended
```

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

```
2181: Function readm: string;  //read last line in memo2 - console!
2182: Function ReadLnWait( AFailCount : Integer) : string
2183: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2184: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2185: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2186: Function ReadString( const ABytes : Integer) : string
2187: Function ReadString( const Section, Ident, Default : string) : string
2188: Function ReadString( Count : Integer) : string
2189: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2190: Function ReadTimeStampCounter : Int64
2191: Function RebootOS : Boolean
2192: Function Receive( ATimeOut : Integer) : TReplyStatus
2193: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2194: Function ReceiveLength : Integer
2195: Function ReceiveText : string
2196: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2197: Function ReceiveSerialText: string
2198: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2199: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
      AMilliSec:Word):TDateTime
2200: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2201: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2202: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2203: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2204: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2205: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2206: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2207: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2208: Function Reconcile( const Results : OleVariant) : Boolean
2209: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2210: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2211: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2212: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2213: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2214: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2215: Function RectCenter( const R : TRect) : TPoint
2216: Function RectEqual( const R1, R2 : TRect) : Boolean
2217: Function RectHeight( const R : TRect) : Integer
2218: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2219: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2220: Function RectIntersection( const R1, R2 : TRect) : TRect
2221: Function RectIntersectRect( const R1, R2 : TRect) : Boolean
2222: Function RectIsEmpty( const R : TRect) : Boolean
2223: Function RectIsNull( const R : TRect) : Boolean
2224: Function RectIsSquare( const R : TRect) : Boolean
2225: Function RectIsValid( const R : TRect) : Boolean
2226: Function RectsAreValid( R : array of TRect) : Boolean
2227: Function RectUnion( const R1, R2 : TRect) : TRect
2228: Function RectWidth( const R : TRect) : Integer
2229: Function RedComponent( const Color32 : TColor32) : Integer
2230: Function Refresh : Boolean
2231: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2232: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2233: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2234: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType
2235: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2236: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2237: Function ReleaseHandle : HBITMAP
2238: Function ReleaseHandle : HENHMETAFILE
2239: Function ReleaseHandle : HICON
2240: Function ReleasePalette : HPALETTE
2241: Function RemainderFloat( const X, Y : Float) : Float
2242: Function Remove( AClass : TClass) : Integer
2243: Function Remove( AComponent : TComponent) : Integer
2244: Function Remove( AItem : Integer) : Integer
2245: Function Remove( AItem : Pointer) : Pointer
2246: Function Remove( AItem : TObject) : TObject
2247: Function Remove( AObject : TObject) : Integer
2248: Function RemoveBackslash( const PathName : string) : string
2249: Function RemoveDF( aString : String) : String  //removes thousand separator
2250: Function RemoveDir( Dir : string) : Boolean
2251: function RemoveDir(const Dir: string): Boolean)
2252: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2253: Function RemoveFileExt( const FileName : string) : string
2254: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2255: Function RenameFile( OldName, NewName : string) : Boolean
2256: function RenameFile(const OldName: string; const NewName: string): Boolean)
2257: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2258: Function ReplaceText( const AText, AFromText, AToText : string) : string
2259: Function Replicate(c : char;I : longInt) : String;
2260: Function Request : TWebRequest
2261: Function ResemblesText( const AText, AOther : string) : Boolean
2262: Function Reset : Boolean
2263: function Reset2(mypath: string):string;
2264: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2265: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2266: Function Response : TWebResponse
2267: Function ResumeSupported : Boolean
2268: Function RETHINKHOTKEYS : BOOLEAN
```

```
2269: Function RETHINKLINES : BOOLEAN
2270: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2271: Function RetrieveCurrentDir : string
2272: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2273: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2274: Function RetrieveMailBoxSize : integer
2275: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2276: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2277: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2278: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2279: Function ReverseBits( Value : Byte) : Byte;
2280: Function ReverseBits1( Value : Shortint) : Shortint;
2281: Function ReverseBits2( Value : Smallint) : Smallint;
2282: Function ReverseBits3( Value : Word) : Word;
2283: Function ReverseBits4( Value : Cardinal) : Cardinal;
2284: Function ReverseBits4( Value : Integer) : Integer;
2285: Function ReverseBits5( Value : Int64) : Int64;
2286: Function ReverseBytes( Value : Word) : Word;
2287: Function ReverseBytes1( Value : Smallint) : Smallint;
2288: Function ReverseBytes2( Value : Integer) : Integer;
2289: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2290: Function ReverseBytes4( Value : Int64) : Int64;
2291: Function ReverseString( const AText : string) : string
2292: Function ReverseDNSLookup(const IPAddrs:String;DNSServer:String;Timeout,Retries:Int;var
      HostName:String):Bool;
2293: Function Revert : HResult
2294: Function RGB(R,G,B: Byte): TColor;
2295: Function RGB2BGR( const Color : TColor) : TColor
2296: Function RGB2TColor( R, G, B : Byte) : TColor
2297: Function RGBToWebColorName( RGB : Integer) : string
2298: Function RGBToWebColorStr( RGB : Integer) : string
2299: Function RgbToHtml( Value : TColor) : string
2300: Function HtmlToRgb(const Value: string): TColor;
2301: Function RightStr( const AStr : String; Len : Integer) : String
2302: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2303: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2304: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2305: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
2306: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2307: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2308: Function Round(e : Extended) : Longint;
2309: Function Round64(e: extended): Int64;
2310: Function RoundAt( const Value : string; Position : SmallInt) : string
2311: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2312: Function RoundTo(const AValue: Extended; const ADigit: TRoundToEXRangeExtended): Extended;');
2313: Function SimpleRoundTo(const AValue: Extended; const ADigit: TRoundToRange): Extended;');
2314: Function RoundFrequency( const Frequency : Integer) : Integer
2315: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2316: Function RoundPoint( const X, Y : Double) : TPoint
2317: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2318: Function RowCount : Integer
2319: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2320: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2321: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2322: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2323: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2324: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2325: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2326: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2327: Function S_AddBackSlash( const ADirName : string) : string
2328: Function S_AllTrim( const cStr : string) : string
2329: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2330: Function S_Cut( const cStr : string; const iLen : integer) : string
2331: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2332: Function S_DirExists( const ADir : string) : Boolean
2333: Function S_Empty( const cStr : string) : boolean
2334: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2335: Function S_LargeFontsActive : Boolean
2336: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2337: Function S_LTrim( const cStr : string) : string
2338: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2339: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2340: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2341: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2342: Function S_RTrim( const cStr : string) : string
2343: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2344: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2345: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2346: Function S_Space( const iLen : integer) : String
2347: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2348: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2349: Function S_StrCRC32( const Text : string) : LongWORD
2350: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2351: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2352: Function S_StringtoUTF_8( const AString : string) : string
2353: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2354: function S_StrToReal(const cStr: string; var R: Double): Boolean
2355: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2356: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
```

```
2357: Function S_UTF_8ToString( const AString : string) : string
2358: Function S_WBox( const AText : string) : integer
2359: Function SameDate( const A, B : TDateTime) : Boolean
2360: function SameDate(const A, B: TDateTime): Boolean;
2361: Function SameDateTime( const A, B : TDateTime) : Boolean
2362: function SameDateTime(const A, B: TDateTime): Boolean;
2363: Function SameFileName( S1, S2 : string) : Boolean
2364: Function SameText( S1, S2 : string) : Boolean
2365: function SameText(const S1: string; const S2: string): Boolean)
2366: Function SameTime( const A, B : TDateTime) : Boolean
2367: function SameTime(const A, B: TDateTime): Boolean;
2368: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2369: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2370: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2371: Function SampleVariance( const X : TDynFloatArray) : Float
2372: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2373: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2374: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2375: Function SaveToFile( const AFileName : TFileName) : Boolean
2376: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2377: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2378: Function ScanF(const aformat: String; const args: array of const): string;
2379: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2380: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
      TStringSearchOptions):PChar
2381: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2382: function SearchRecattr: integer;
2383: function SearchRecExcludeAttr: integer;
2384: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2385: function SearchRecname: string;
2386: function SearchRecsize: integer;
2387: function SearchRecTime: integer;
2388: Function Sec( const X : Extended) : Extended
2389: Function Secant( const X : Extended) : Extended
2390: Function SecH( const X : Extended) : Extended
2391: Function SecondOf( const AValue : TDateTime) : Word
2392: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2393: Function SecondOfTheHour( const AValue : TDateTime) : Word
2394: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2395: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2396: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2397: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2398: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2399: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2400: Function SectionExists( const Section : string) : Boolean
2401: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2402: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2403: function Seek(Offset:LongInt;Origin:Word):LongInt
2404: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2405: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2406: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2407: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2408: Function SendBuf( var Buf, Count : Integer) : Integer
2409: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2410: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2411: Function SendKey( AppName : string; Key : Char) : Boolean
2412: Function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2413: Function SendStream( AStream : TStream) : Boolean
2414: Function SendStreamThenDrop( AStream : TStream) : Boolean
2415: Function SendText( const S : string) : Integer
2416: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2417: Function SendSerialText(Data: String): cardinal
2418: Function Sent : Boolean
2419: Function ServicesFilePath: string
2420: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2421: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2422: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2423: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2424: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2425: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2426: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2427: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2428: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2429: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2430: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2431: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2432: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2433: Function SetCurrentDir( Dir : string) : Boolean
2434: function SetCurrentDir(const Dir: string): Boolean)
2435: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2436: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2437: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2438: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2439: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2440: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2441: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2442: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
```

```
2443: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2444: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2445: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2446: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2447: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2448: Function SetLocalTime( Value : TDateTime) : boolean
2449: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2450: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2451: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2452: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2453: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2454: Function SetSize( libNewSize : Longint) : HResult
2455: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2456: Function Sgn( const X : Extended) : Integer
2457: function SHA1(const fileName: string): string;
2458: function SHA256(astr: string; amode: char): string)
2459: function SHA512(astr: string; amode: char): string)
2460: Function ShareMemoryManager : Boolean
2461: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2462: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2463: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2464: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2465: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2466: function ShortDateFormat: string;
2467: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2468: function ShortTimeFormat: string;
2469: function SHOWMODAL:INTEGER
2470: Function ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:
      TColor; BW: Integer;Title:String;BeforeShowModal:TNotifyEvent): TModalResult');
2471: Function
      ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2472: function ShowWindow(C1: HWND; C2: integer): boolean;
2473: procedure ShowMemory     //in Dialog
2474: function ShowMemory2: string;
2475: Function ShutDownOS : Boolean
2476: Function Signe( const X, Y : Extended) : Extended
2477: Function Sign( const X : Extended) : Integer
2478: Function Sin(e : Extended) : Extended;
2479: Function sinc( const x : Double) : Double
2480: Function SinJ( X : Float) : Float
2481: Function Size( const AFileName : String) : Integer
2482: function SizeOf: Longint;
2483: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2484: function SlashSep(const Path, S: String): String
2485: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2486: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2487: Function SmallPoint(X, Y: Integer): TSmallPoint)
2488: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2489: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2490: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2491: Function SoundexProc( const AText, AOther : string) : Boolean
2492: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2493: Function SoundexWord( const AText : string) : Word
2494: Function SourcePos : Longint
2495: function SourcePos:LongInt
2496: Function Split0( Str : string; const substr : string) : TStringList
2497: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2498: Function SQLRequiresParams( const SQL : WideString) : Boolean
2499: Function Sqr(e : Extended) : Extended;
2500: Function Sqrt(e : Extended) : Extended;
2501: Function StartIP : String
2502: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2503: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2504: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2505: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2506: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2507: Function StartOfAYear( const AYear : Word) : TDateTime
2508: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2509: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2510: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2511: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2512: Function StartsStr( const ASubText, AText : string) : Boolean
2513: Function StartsText( const ASubText, AText : string) : Boolean
2514: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2515: Function StartsWith( const str : string; const sub : string) : Boolean
2516: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2517: Function StatusString( StatusCode : Integer) : string
2518: Function StdDev( const Data : array of Double) : Extended
2519: Function Stop : Float
2520: Function StopCount( var Counter : TJclCounter) : Float
2521: Function StoreColumns : Boolean
2522: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2523: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2524: Function StrAlloc( Size : Cardinal) : PChar
2525: function StrAlloc(Size: Cardinal): PChar)
2526: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2527: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2528: Function StrBufSize( Str : PChar) : Cardinal
```

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

```
2618: function StrToDateTime(const S: string): TDateTime)
2619: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2620: Function StrToDay( const ADay : string) : Byte
2621: Function StrToFloat( S : string) : Extended;
2622: function StrToFloat(s: String): Extended;
2623: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2624: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2625: Function StrToFloat( S : string) : Extended;
2626: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2627: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2628: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2629: Function StrToCurr( S : string) : Currency;
2630: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2631: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2632: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2633: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2634: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2635: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2636: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2637: Function StrToDateTime( S : string) : TDateTime;
2638: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2639: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2640: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2641: Function StrToInt( S : string) : Integer
2642: function StrToInt(s: String): Longint;
2643: Function StrToInt64( S : string) : Int64
2644: function StrToInt64(s: String): int64;
2645: Function StrToInt64Def( S : string; Default : Int64) : Int64
2646: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2647: Function StrToIntDef( S : string; Default : Integer) : Integer
2648: function StrToIntDef(const S: string; Default: Integer): Integer)
2649: function StrToIntDef(s: String; def: Longint): Longint;
2650: Function StrToMonth( const AMonth : string) : Byte
2651: Function StrToTime( S : string) : TDateTime;
2652: function StrToTime(const S: string): TDateTime)
2653: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2654: Function StrToWord( const Value : String) : Word
2655: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2656: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2657: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2658: Function StrUpper( Str : PChar) : PChar
2659: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2660: Function Sum( const Data : array of Double) : Extended
2661: Function SumFloatArray( const B : TDynFloatArray) : Float
2662: Function SumInt( const Data : array of Integer) : Integer
2663: Function SumOfSquares( const Data : array of Double) : Extended
2664: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2665: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2666: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2667: Function Supports( CursorOptions : TCursorOptions) : Boolean
2668: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2669: Function SwapWord(w : word): word)
2670: Function SwapInt(i : integer): integer)
2671: Function SwapLong(L : longint): longint)
2672: Function Swap(i : integer): integer)
2673: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2674: Function SyncTime : Boolean
2675: Function SysErrorMessage( ErrorCode : Integer) : string
2676: function SysErrorMessage(ErrorCode: Integer): string)
2677: Function SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2678: function SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2679: Function SysStringLen(const S: WideString): Integer; stdcall;
2680: Function TabRect( Index : Integer) : TRect
2681: Function Tan( const X : Extended) : Extended
2682: Function TaskMessageDlg(const Title,
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2683: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2684: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2685: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2686: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2687: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2688: Function TenToY( const Y : Float) : Float
2689: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2690: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2691: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2692: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2693: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2694: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2695: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2696: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2697: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2698: Function TestBits( const Value, Mask : Byte) : Boolean;
2699: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2700: Function TestBits2( const Value, Mask : Smallint) : Boolean;
```

```
2701: Function TestBits3( const Value, Mask : Word) : Boolean;
2702: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2703: Function TestBits5( const Value, Mask : Integer) : Boolean;
2704: Function TestBits6( const Value, Mask : Int64) : Boolean;
2705: Function TestFDIVInstruction : Boolean
2706: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2707: Function TextExtent( const Text : string) : TSize
2708: function TextHeight(Text: string): Integer;
2709: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2710: Function TextStartsWith( const S, SubS : string) : Boolean
2711: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2712: Function ConvInteger(i : integer):string;
2713: Function IntegerToText(i : integer):string;
2714: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2715: function TextWidth(Text: string): Integer;
2716: Function ThreadCount : integer
2717: function ThousandSeparator: char;
2718: Function Ticks : Cardinal
2719: Function Time : TDateTime
2720: function Time: TDateTime;
2721: function TimeGetTime: int64;
2722: Function TimeOf( const AValue : TDateTime) : TDateTime
2723: function TimeSeparator: char;
2724: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2725: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2726: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2727: Function TimeToStr( DateTime : TDateTime) : string;
2728: function TimeToStr(const DateTime: TDateTime): string;
2729: Function TimeZoneBias : TDateTime
2730: Function ToCommon( const AValue : Double) : Double
2731: function ToCommon(const AValue: Double): Double;
2732: Function Today : TDateTime
2733: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2734: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2735: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2736: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2737: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2738: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2739: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2740: function TokenComponentIdent:String
2741: Function TokenFloat : Extended
2742: function TokenFloat:Extended
2743: Function TokenInt : Longint
2744: function TokenInt:LongInt
2745: Function TokenString : string
2746: function TokenString:String
2747: Function TokenSymbolIs( const S : string) : Boolean
2748: function TokenSymbolIs(S:String):Boolean
2749: Function Tomorrow : TDateTime
2750: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2751: Function ToString : string
2752: Function TotalVariance( const Data : array of Double) : Extended
2753: Function Trace2( AURL : string) : string;
2754: Function TrackMenu( Button : TToolButton) : Boolean
2755: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2756: Function TranslateURI( const URI : string) : string
2757: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2758: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2759: Function Trim( S : string) : string;
2760: Function Trim( S : WideString) : WideString;
2761: Function Trim(s : AnyString) : AnyString;
2762: Function TrimAllOf( ATrim, AText : String) : String
2763: Function TrimLeft( S : string) : string;
2764: Function TrimLeft( S : WideString) : WideString;
2765: function TrimLeft(const S: string): string)
2766: Function TrimRight( S : string) : string;
2767: Function TrimRight( S : WideString) : WideString;
2768: function TrimRight(const S: string): string)
2769: function TrueBoolStrs: array of string
2770: Function Trunc(e : Extended) : Longint;
2771: Function Trunc64(e: extended): Int64;
2772: Function TruncPower( const Base, Exponent : Float) : Float
2773: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2774: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2775: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2776: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2777: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2778: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
      AValue:TDateTime):Boolean
2779: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2780: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
      AVal:TDateTime):Bool
2781: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2782: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2783: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2784: Function TryLock : Boolean
2785: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2786: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word; out AResult : TDateTime) : Boolean
```

```
2787: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2788: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2789: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2790: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2791: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2792: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2793: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2794: Function TryStrToBool(const S: string; out Value: Boolean): Boolean;
2795: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2796: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2797: Function TwoToY( const Y : Float) : Float
2798: Function UCS4StringToWideString( const S : UCS4String) : WideString
2799: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2800: function Unassigned: Variant;
2801: Function UndoLastChange( FollowChange : Boolean) : Boolean
2802: function UniCodeToStr(Value: string): string;
2803: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2804: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2805: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2806: Function UnixPathToDosPath( const Path : string) : string
2807: Function UnixToDateTime( const AValue : Int64) : TDateTime
2808: function UnixToDateTime(U: Int64): TDateTime;
2809: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2810: Function UnlockResource( ResData : HGLOBAL) : LongBool
2811: Function UnlockVolume( var Handle : THandle) : Boolean
2812: Function UnMaskString( Mask, Value : String) : String
2813: function UpCase(ch : Char ) : Char;
2814: Function UpCaseFirst( const AStr : string) : string
2815: Function UpCaseFirstWord( const AStr : string) : string
2816: Function UpdateAction( Action : TBasicAction) : Boolean
2817: Function UpdateKind : TUpdateKind
2818: Function UPDATESTATUS : TUPDATESTATUS
2819: Function UpperCase( S : string) : string
2820: Function Uppercase(s : AnyString) : AnyString;
2821: Function URLDecode( ASrc : string) : string
2822: Function URLEncode( const ASrc : string) : string
2823: Function UseRightToLeftAlignment : Boolean
2824: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2825: Function UseRightToLeftReading : Boolean
2826: Function UTF8CharLength( Lead : Char) : Integer
2827: Function UTF8CharSize( Lead : Char) : Integer
2828: Function UTF8Decode( const S : UTF8String) : WideString
2829: Function UTF8Encode( const WS : WideString) : UTF8String
2830: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2831: Function Utf8ToAnsi( const S : UTF8String) : string
2832: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2833: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2834: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2835: Function ValidParentForm(control: TControl): TForm
2836: Function Value : Variant
2837: Function ValueExists( const Section, Ident : string) : Boolean
2838: Function ValueOf( const Key : string) : Integer
2839: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2840: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2841: Function VarArrayFromStrings( Strings : TStrings) : Variant
2842: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2843: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2844: Function VarFMTBcd : TVarType
2845: Function VarFMTBcdCreate1 : Variant;
2846: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2847: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2848: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2849: Function Variance( const Data : array of Double) : Extended
2850: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2851: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2852: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2853: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2854: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2855: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2856: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2857: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2858: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2859: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2860: Function VariantNeg( const V1 : Variant) : Variant
2861: Function VariantNot( const V1 : Variant) : Variant
2862: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2863: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2864: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2865: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2866: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2867: function VarIsEmpty(const V: Variant): Boolean;
2868: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2869: function VarIsNull(const V: Variant): Boolean;
2870: Function VarToBcd( const AValue : Variant) : TBcd
2871: function VarType(const V: Variant): TVarType;
2872: Function VarType( const V : Variant) : TVarType
2873: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2874: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2875: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
```

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

```
2965: Function WideCopy( const src : WideString; index, count : Integer) : WideString
2966: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
2967: Function WideEqual( const x, y : WideString) : Boolean
2968: function WideFormat(const Format: WideString; const Args: array of const): WideString)
2969: Function WideGreater( const x, y : WideString) : Boolean
2970: Function WideLength( const src : WideString) : Integer
2971: Function WideLess( const x, y : WideString) : Boolean
2972: Function WideLowerCase( S : WideString) : WideString
2973: function WideLowerCase(const S: WideString): WideString)
2974: Function WidePos( const src, sub : WideString) : Integer
2975: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
2976: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
2977: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
2978: Function WideSameStr( S1, S2 : WideString) : Boolean
2979: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
2980: Function WideSameText( S1, S2 : WideString) : Boolean
2981: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
2982: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
2983: Function WideStringToUCS4String( const S : WideString) : UCS4String
2984: Function WideUpperCase( S : WideString) : WideString
2985: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
2986: function Win32Check(RetVal: boolean): boolean)
2987: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
2988: Function Win32RestoreFile( const FileName : string) : Boolean
2989: Function Win32Type : TIdWin32Type
2990: Function WinColor( const Color32 : TColor32) : TColor
2991: function winexec(FileName: pchar; showCmd: integer): integer;
2992: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
2993: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
2994: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
2995: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
2996: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
2997: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
2998: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
2999: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
3000: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
3001: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
3002: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
3003: Function WordToStr( const Value : Word) : String
3004: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
3005: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
3006: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
3007: Function WorkArea : Integer
3008: Function WrapText( Line : string; MaxCol : Integer) : string;
3009: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
3010: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
3011: function Write(Buffer:String;Count:LongInt):LongInt
3012: Function WriteClient( var Buffer, Count : Integer) : Integer
3013: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
3014: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
3015: Function WriteString( const AString : string) : Boolean
3016: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
3017: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
3018: Function wsprintf( Output : PChar; Format : PChar) : Integer
3019: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
3020: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
3021: Function XorDecode( const Key, Source : string) : string
3022: Function XorEncode( const Key, Source : string) : string
3023: Function XorString( const Key, Src : ShortString) : ShortString
3024: Function Yield : Bool
3025: Function YearOf( const AValue : TDateTime) : Word
3026: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3027: Function YearSpan( const ANow, AThen : TDateTime) : Double
3028: Function Yesterday : TDateTime
3029: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3030: Function( const Name : string; Proc : TUserFunction)
3031: Function using Special_Scholz from 3.8.5.0
3032: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3033: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3034: Function FloatToTime2Dec(value:Extended):Extended;
3035: Function MinToStd(value:Extended):Extended;
3036: Function MinToStdAsString(value:Extended):String;
3037: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3038: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3039: Function Round2Dec (zahl:Extended):Extended;
3040: Function GetAngle(x,y:Extended):Double;
3041: Function AddAngle(a1,a2:Double):Double;
3042:
3043: ************************************************************************
3044: unit uPSI_StText;
3045: ************************************************************************
3046: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3047: Function TextFileSize( var F : TextFile) : LongInt
3048: Function TextPos( var F : TextFile) : LongInt
3049: Function TextFlush( var F : TextFile) : Boolean
3050:
3051: ************************************************************************
3052: from  JvVCLUtils;
3053: ************************************************************************
```

```
3054: { Windows resources (bitmaps and icons) VCL-oriented routines }
3055: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3056: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
      DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3057: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
      Bitmap: TBitmap; TransparentColor: TColor);
3058: function MakeBitmap(ResID: PChar): TBitmap;
3059: function MakeBitmapID(ResID: Word): TBitmap;
3060: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3061: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3062: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3063: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3064:   HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3065: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3066: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3067: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3068: {$IFDEF WIN32}
3069: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3070:   X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3071: {$ENDIF}
3072: function MakeIcon(ResID: PChar): TIcon;
3073: function MakeIconID(ResID: Word): TIcon;
3074: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3075: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3076: {$IFDEF WIN32}
3077: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3078: {$ENDIF}
3079: { Service routines }
3080: procedure NotImplemented;
3081: procedure ResourceNotFound(ResID: PChar);
3082: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3083: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3084: function PaletteColor(Color: TColor): Longint;
3085: function WidthOf(R: TRect): Integer;
3086: function HeightOf(R: TRect): Integer;
3087: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3088: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3089: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3090: procedure Delay(MSecs: Longint);
3091: procedure CenterControl(Control: TControl);
3092: Function PaletteEntries( Palette : HPALETTE) : Integer
3093: Function WindowClassName( Wnd : HWND) : string
3094: Function ScreenWorkArea : TRect
3095: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3096: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3097: Procedure ActivateWindow( Wnd : HWND)
3098: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3099: Procedure CenterWindow( Wnd : HWND)
3100: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3101: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3102: Function DialogsToPixelsX( Dlgs : Word) : Word
3103: Function DialogsToPixelsY( Dlgs : Word) : Word
3104: Function PixelsToDialogsX( Pixs : Word) : Word
3105: Function PixelsToDialogsY( Pixs : Word) : Word
3106: {$IFDEF WIN32}
3107: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3108: function MakeVariant(const Values: array of Variant): Variant;
3109: {$ENDIF}
3110: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3111: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3112: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3113: {$IFDEF CBUILDER}
3114: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3115: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3116: {$ELSE}
3117: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3118: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3119: {$ENDIF CBUILDER}
3120: function IsForegroundTask: Boolean;
3121: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3122: function GetAveCharSize(Canvas: TCanvas): TPoint;
3123: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3124: procedure FreeUnusedOle;
3125: procedure Beep;
3126: function GetWindowsVersionJ: string;
3127: function LoadDLL(const LibName: string): THandle;
3128: function RegisterServer(const ModuleName: string): Boolean;
3129: {$IFNDEF WIN32}
3130: function IsLibrary: Boolean;
3131: {$ENDIF}
3132: { Gradient filling routine }
3133: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3134: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
      TFillDirection; Colors: Byte);
3135: { String routines }
3136: function GetEnvVar(const VarName: string): string;
3137: function AnsiUpperFirstChar(const S: string): string;
3138: function StringToPChar(var S: string): PChar;
3139: function StrPAlloc(const S: string): PChar;
```

```
3140: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3141: function DropT(const S: string): string;
3142: { Memory routines }
3143: function AllocMemo(Size: Longint): Pointer;
3144: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3145: procedure FreeMemo(var fpBlock: Pointer);
3146: function GetMemoSize(fpBlock: Pointer): Longint;
3147: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3148: {$IFNDEF COMPILER5_UP}
3149: procedure FreeAndNil(var Obj);
3150: {$ENDIF}
3151: // from PNGLoader
3152: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3153: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3154: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3155: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3156: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
      TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
3157:   Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3158:   Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3159:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3160:  //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3161:  //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3162:  //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3163:  Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3164:  Procedure SetImeName( Name : TImeName)
3165:  Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3166:  Function Imm32GetContext( hWnd : HWND) : HIMC
3167:  Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
3168:  Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
3169:  Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3170:  Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3171: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3172:  //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
3173:  Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3174:  Function Imm32IsIME( hKl : longword) : Boolean
3175:  Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3176:  Procedure DragDone( Drop : Boolean)
3177:
3178:
3179: //**************************************added  from jvjvclutils
3180: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3181: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3182: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3183: function IsPositiveResult(Value: TModalResult): Boolean;
3184: function IsNegativeResult(Value: TModalResult): Boolean;
3185: function IsAbortResult(const Value: TModalResult): Boolean;
3186: function StripAllFromResult(const Value: TModalResult): TModalResult;
3187: // returns either BrightColor or DarkColor depending on the luminance of AColor
3188: // This function gives the same result (AFAIK) as the function used in Windows to
3189: // calculate the desktop icon text color based on the desktop background color
3190: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3191: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3192:
3193: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3194:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3195:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3196:   var LinkName: string; Scale: Integer = 100); overload;
3197: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3198:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3199:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3200:   var LinkName: string; Scale: Integer = 100); overload;
3201: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3202:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3203: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3204:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3205:   Scale: Integer = 100): string;
3206: function HTMLPlainText(const Text: string): string;
3207: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3208:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3209: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3210:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3211: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3212: function HTMLPrepareText(const Text: string): string;
3213:
3214: ************************************** uPSI_JvAppUtils;
3215: Function GetDefaultSection( Component : TComponent) : string
3216: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3217: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3218: Function GetDefaultIniName : string
3219:  //'OnGetDefaultIniName','TOnGetDefaultIniName);
3220: Function GetDefaultIniRegKey : string
3221: Function FindForm( FormClass : TFormClass) : TForm
3222: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3223: Function ShowDialog( FormClass : TFormClass) : Boolean
3224:  //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3225: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3226: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3227: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
```

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

```
3315:  Function HourOfTime( const DateTime : TDateTime) : Integer
3316:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3317:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3318:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3319:  Function IsISOLongYear( const Year : Word) : Boolean;
3320:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3321:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3322:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3323:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3324:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3325:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3326:  Function JIsLeapYear( const Year : Integer) : Boolean;
3327:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3328:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3329:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3330:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3331:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3332:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3333:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3334:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3335:  Function HoursToMSecs( Hours : Integer) : Integer
3336:  Function MinutesToMSecs( Minutes : Integer) : Integer
3337:  Function SecondsToMSecs( Seconds : Integer) : Integer
3338:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3339:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3340:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3341:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3342:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3343:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3344:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3345:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3346:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3347:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3348:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3349:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3350:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3351:  Function DosDateTimeToStr( DateTime : Integer) : string
3352:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3353:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3354:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3355:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3356:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3357:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3358:  Function FileTimeToStr( const FileTime : TFileTime) : string
3359:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3360:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3361:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3362:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3363:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3364:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3365:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3366:   TJclUnixTime32', 'Longword
3367:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3368:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3369:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3370:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3371:  Function JNullStamp : TTimeStamp
3372:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3373:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3374:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3375:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3376:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3377:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3378:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3379:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3380:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3381:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3382:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3383:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3384:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3385:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3386:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3387:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3388:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3389:   FindClass('TOBJECT'),'EJclDateTimeError
3390: end;
3391:
3392: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3393: begin
3394:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3395:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3396:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3397:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3398:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3399:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3400:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3401:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3402:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3403:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
```

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

```
3492:  Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3493:  Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3494:  Function StrStrCount( const S, SubS : AnsiString) : Integer
3495:  Function StrCompare( const S1, S2 : AnsiString) : Integer
3496:  Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3497:  //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3498:  Function StrFillChar1( const C : Char; Count : Integer) : AnsiString;
3499:  Function StrFillChar(const C: Char; Count: Integer): string)');
3500:  Function IntFillChar(const I: Integer; Count: Integer): string)');
3501:  Function ByteFillChar(const B: Byte; Count: Integer): string)');
3502:  Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;');
3503:  Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3504:  Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3505:  //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3506:  Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3507:  Function StrILastPos( const SubStr, S : AnsiString) : Integer
3508:  Function StrIPos( const SubStr, S : AnsiString) : Integer
3509:  Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString) : Boolean
3510:  Function StrLastPos( const SubStr, S : AnsiString) : Integer
3511:  Function StrMatch( const Substr, S : AnsiString; const Index : Integer) : Integer
3512:  Function StrMatches( const Substr, S : AnsiString; const Index : Integer) : Boolean
3513:  Function StrNIPos( const S, SubStr : AnsiString; N : Integer) : Integer
3514:  Function StrNPos( const S, SubStr : AnsiString; N : Integer) : Integer
3515:  Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString) : Integer
3516:  Function StrSearch( const Substr, S : AnsiString; const Index : Integer) : Integer
3517:  //Function StrAfter( const SubStr, S : AnsiString) : AnsiString
3518:  //Function StrBefore( const SubStr, S : AnsiString) : AnsiString
3519:  Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar) : AnsiString
3520:  Function StrChopRight( const S : AnsiString; N : Integer) : AnsiString
3521:  Function StrLeft( const S : AnsiString; Count : Integer) : AnsiString
3522:  Function StrMid( const S : AnsiString; Start, Count : Integer) : AnsiString
3523:  Function StrRestOf( const S : AnsiString; N : Integer) : AnsiString
3524:  Function StrRight( const S : AnsiString; Count : Integer) : AnsiString
3525:  Function CharEqualNoCase( const C1, C2 : AnsiChar) : Boolean
3526:  Function CharIsAlpha( const C : AnsiChar) : Boolean
3527:  Function CharIsAlphaNum( const C : AnsiChar) : Boolean
3528:  Function CharIsBlank( const C : AnsiChar) : Boolean
3529:  Function CharIsControl( const C : AnsiChar) : Boolean
3530:  Function CharIsDelete( const C : AnsiChar) : Boolean
3531:  Function CharIsDigit( const C : AnsiChar) : Boolean
3532:  Function CharIsLower( const C : AnsiChar) : Boolean
3533:  Function CharIsNumberChar( const C : AnsiChar) : Boolean
3534:  Function CharIsPrintable( const C : AnsiChar) : Boolean
3535:  Function CharIsPunctuation( const C : AnsiChar) : Boolean
3536:  Function CharIsReturn( const C : AnsiChar) : Boolean
3537:  Function CharIsSpace( const C : AnsiChar) : Boolean
3538:  Function CharIsUpper( const C : AnsiChar) : Boolean
3539:  Function CharIsWhiteSpace( const C : AnsiChar) : Boolean
3540:  Function CharType( const C : AnsiChar) : Word
3541:  Function CharHex( const C : AnsiChar) : Byte
3542:  Function CharLower( const C : AnsiChar) : AnsiChar
3543:  Function CharUpper( const C : AnsiChar) : AnsiChar
3544:  Function CharToggleCase( const C : AnsiChar) : AnsiChar
3545:  Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3546:  Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3547:  Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer) : Integer
3548:  Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar) : Integer
3549:  Procedure StrIToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3550:  Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3551:  Function StringsToStr(const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool):AnsiString;
3552:  Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean)
3553:  Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean)
3554:  Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean)
3555:  Function AddStringToStrings(const S:AnsiString;Strings:TStrings; const Unique:Boolean):Boolean
3556:  Function BooleanToStr( B : Boolean) : AnsiString
3557:  Function FileToString( const FileName : AnsiString) : AnsiString
3558:  Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean)
3559:  Function StrToken( var S : AnsiString; Separator : AnsiChar) : AnsiString
3560:  Procedure StrTokens( const S : AnsiString; const List : TStrings)
3561:  Procedure StrTokenToStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings)
3562:  //Function StrWord( var S : PAnsiChar; out Word : AnsiString) : Boolean
3563:  Function StrToFloatSafe( const S : AnsiString) : Float
3564:  Function StrToIntSafe( const S : AnsiString) : Integer
3565:  Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer);
3566:  Function ArrayOf( List : TStrings) : TDynStringArray;
3567:   EJclStringError', 'EJclError
3568:  function IsClass(Address: TObject): Boolean;
3569:  function IsObject(Address: TObject): Boolean;
3570:  // Console Utilities
3571:  //function ReadKey: Char;
3572:  function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3573:  function JclGUIDToString(const GUID: TGUID): string;
3574:  function JclStringToGUID(const S: string): TGUID;
3575:
3576: end;
3577:
3578:
3579: ****************************************** uPSI_JvDBUtil;
3580: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const
     Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
```

```
3581: Function GetQueryResult( const DatabaseName, SQL : string) : Variant
3582: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant;
      const AResultName : string) : Variant
3583: //Function StrFieldDesc( Field : FLDDesc) : string
3584: Function Var2Type( V : Variant; const VarType : Integer) : Variant
3585: Procedure CopyRecord( DataSet : TDataSet)
3586: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string;
      MasterField : Word; ModOp, DelOp : RINTQual)
3587: Procedure AddMasterPassword( Table : TTable; pswd : string)
3588: Procedure PackTable( Table : TTable)
3589: Procedure PackEncryptedTable( Table : TTable; pswd : string)
3590: Function EncodeQuotes( const S : string) : string
3591: Function Cmp( const S1, S2 : string) : Boolean
3592: Function SubStr( const S : string; const Index : Integer; const Separator : string) : string
3593: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string) : string
3594: Function ReplaceString( S : string; const OldPattern, NewPattern : string) : string
3595: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer)
3596: ******************************************uPSI_JvJvBDEUtils;**************
3597: //JvBDEUtils
3598: Function CreateDbLocate : TJvLocateObject
3599:  //Function CheckOpen( Status : DBIResult) : Boolean
3600: Procedure FetchAllRecords( DataSet : TBDEDataSet)
3601: Function TransActive( Database : TDatabase) : Boolean
3602: Function AsyncQrySupported( Database : TDatabase) : Boolean
3603: Function GetQuoteChar( Database : TDatabase) : string
3604: Procedure ExecuteQuery( const DbName, QueryText : string)
3605: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string)
3606: Function FieldLogicMap( FldType : TFieldType) : Integer
3607: Function FieldSubtypeMap( FldType : TFieldType) : Integer Value : string; Buffer : Pointer)
3608: Function GetAliasPath( const AliasName : string) : string
3609: Function IsDirectory( const DatabaseName : string) : Boolean
3610: Function GetBdeDirectory : string
3611: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) : Boolean
3612: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3613: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3614: Function DataSetRecNo( DataSet : TDataSet) : Longint
3615: Function DataSetRecordCount( DataSet : TDataSet) : Longint
3616: Function DataSetPositionStr( DataSet : TDataSet) : string
3617: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean)
3618: Function CurrentRecordDeleted( DataSet : TBDEDataSet) : Boolean
3619: Function IsFilterApplicable( DataSet : TDataSet) : Boolean
3620: Function IsBookmarkStable( DataSet : TBDEDataSet) : Boolean
3621: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)
3622: Procedure RestoreIndex( Table : TTable)
3623: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)
3624: Procedure PackTable( Table : TTable)
3625: Procedure ReindexTable( Table : TTable)
3626: Procedure BdeFlushBuffers
3627: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer) : Pointer
3628: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3629: Procedure DbNotSupported
3630: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3631: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3632: Procedure
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3633: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3634: ******************************************uPSI_JvDateUtil;
3635: function CurrentYear: Word;
3636: function IsLeapYear(AYear: Integer): Boolean;
3637: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3638: function FirstDayOfPrevMonth: TDateTime;
3639: function LastDayOfPrevMonth: TDateTime;
3640: function FirstDayOfNextMonth: TDateTime;
3641: function ExtractDay(ADate: TDateTime): Word;
3642: function ExtractMonth(ADate: TDateTime): Word;
3643: function ExtractYear(ADate: TDateTime): Word;
3644: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3645: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3646: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3647: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3648: function ValidDate(ADate: TDateTime): Boolean;
3649: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3650: function MonthsBetween(Date1, Date2: TDateTime): Double;
3651: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3652: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3653: function DaysBetween(Date1, Date2: TDateTime): Longint;
3654: { The same as previous but if Date2 < Date1 result = 0 }
3655: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3656: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3657: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3658: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3659: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3660: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3661: { String to date conversions }
3662: function GetDateOrder(const DateFormat: string): TDateOrder;
3663: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3664: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
```

```
3665: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3666: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3667: function DefDateFormat(FourDigitYear: Boolean): string;
3668: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3669: ------------------------------------------------------------------------------
3670: ******************************** JvUtils;*****************************
3671: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3672: function GetWordOnPos(const S: string; const P: Integer): string;
3673: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3674: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3675: { SubStr returns substring from string, S, separated with Separator string}
3676: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3677: { SubStrEnd same to previous function but Index numerated from the end of string }
3678: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3679: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3680: function SubWord(P: PChar; var P2: PChar): string;
3681: { NumberByWord returns the text representation of
3682:   the number, N, in normal russian language. Was typed from Monitor magazine }
3683: function NumberByWord(const N: Longint): string;
3684: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3685:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3686: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3687: { GetXYByPos is same to previous function, but returns X position in line too}
3688: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3689: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3690: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3691: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3692: function ConcatSep(const S, S2, Separator: string): string;
3693: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3694: function ConcatLeftSep(const S, S2, Separator: string): string;
3695: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3696: function MinimizeString(const S: string; const MaxLen: Integer): string;
3697: { Next 4 function for russian chars transliterating.
3698:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3699: procedure Dos2Win(var S: string);
3700: procedure Win2Dos(var S: string);
3701: function Dos2WinRes(const S: string): string;
3702: function Win2DosRes(const S: string): string;
3703: function Win2Koi(const S: string): string;
3704: { Spaces returns string consists on N space chars }
3705: function Spaces(const N: Integer): string;
3706: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3707: function AddSpaces(const S: string; const N: Integer): string;
3708: { function LastDate for russian users only } { returns date relative to current date: '' }
3709: function LastDate(const Dat: TDateTime): string;
3710: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3711: function CurrencyToStr(const Cur: currency): string;
3712: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3713: function Cmp(const S1, S2: string): Boolean;
3714: { StringCat add S2 string to S1 and returns this string }
3715: function StringCat(var S1: string; S2: string): string;
3716: { HasChar returns True, if Char, Ch, contains in string, S }
3717: function HasChar(const Ch: Char; const S: string): Boolean;
3718: function HasAnyChar(const Chars: string; const S: string): Boolean;
3719: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3720: function CountOfChar(const Ch: Char; const S: string): Integer;
3721: function DefStr(const S: string; Default: string): string;
3722: {**** files routines}
3723: { GetWinDir returns Windows folder name }
3724: function GetWinDir: TFileName;
3725: function GetSysDir: String;
3726: { GetTempDir returns Windows temporary folder name }
3727: function GetTempDir: string;
3728: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3729: function GenTempFileName(FileName: string): string;
3730: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3731: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3732: { ClearDir clears folder Dir }
3733: function ClearDir(const Dir: string): Boolean;
3734: { DeleteDir clears and than delete folder Dir }
3735: function DeleteDir(const Dir: string): Boolean;
3736: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3737: function FileEquMask(FileName, Mask: TFileName): Boolean;
3738: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3739:   Masks must be separated with comma (';') }
3740: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3741: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3742: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3743: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3744: { FileGetInfo fills SearchRec record for specified file attributes}
3745: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3746: { HasSubFolder returns True, if folder APath contains other folders }
3747: function HasSubFolder(APath: TFileName): Boolean;
3748: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3749: function IsEmptyFolder(APath: TFileName): Boolean;
3750: { AddSlash add slash Char to Dir parameter, if needed }
3751: procedure AddSlash(var Dir: TFileName);
3752: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3753: function AddSlash2(const Dir: TFileName): string;
```

```
3754: { AddPath returns FileName with Path, if FileName not contain any path }
3755: function AddPath(const FileName, Path: TFileName): TFileName;
3756: function AddPaths(const PathList, Path: string): string;
3757: function ParentPath(const Path: TFileName): TFileName;
3758: function FindInPath(const FileName, PathList: string): TFileName;
3759: function FindInPaths(const fileName,paths: String): String;
3760: {$IFNDEF BCB1}
3761: { BrowseForFolder displays Browse For Folder dialog }
3762: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3763: {$ENDIF BCB1}
3764: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3765: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3766: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3767: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3768:
3769: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3770: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3771: { HasParam returns True, if program running with specified parameter, Param }
3772: function HasParam(const Param: string): Boolean;
3773: function HasSwitch(const Param: string): Boolean;
3774: function Switch(const Param: string): string;
3775: { ExePath returns ExtractFilePath(ParamStr(0)) }
3776: function ExePath: TFileName;
3777: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3778: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3779: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3780: {**** Graphic routines }
3781: { TTFontSelected returns True, if True Type font is selected in specified device context }
3782: function TTFontSelected(const DC: HDC): Boolean;
3783: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3784: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3785: {**** Windows routines }
3786: { SetWindowTop put window to top without recreating window }
3787: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3788: {**** other routines }
3789: { KeyPressed returns True, if Key VK is now pressed }
3790: function KeyPressed(VK: Integer): Boolean;
3791: procedure SwapInt(var Int1, Int2: Integer);
3792: function IntPower(Base, Exponent: Integer): Integer;
3793: function ChangeTopException(E: TObject): TObject;
3794: function StrToBool(const S: string): Boolean;
3795: {$IFNDEF COMPILER3_UP}
3796: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3797:   Length of MaxLen bytes. The compare operation is controlled by the
3798:   current Windows locale. The return value is the same as for CompareStr. }
3799: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3800: function AnsiStrIComp(S1, S2: PChar): Integer;
3801: {$ENDIF}
3802: function Var2Type(V: Variant; const VarType: Integer): Variant;
3803: function VarToInt(V: Variant): Integer;
3804: function VarToFloat(V: Variant): Double;
3805: { following functions are not documented because they are don't work properly , so don't use them }
3806: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3807: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3808: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3809: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3810: function GetParameter: string;
3811: function GetLongFileName(FileName: string): string;
3812: {* from  FileCtrl}
3813: function DirectoryExists(const Name: string): Boolean;
3814: procedure ForceDirectories(Dir: string);
3815: {# from  FileCtrl}
3816: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3817: function GetComputerID: string;
3818: function GetComputerName: string;
3819: {**** string routines }
3820: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
     same Index.Also see RAUtilsW.ReplaceSokr1 function }
3821: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3822: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3823:   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 }
3824: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3825: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3826: function CountOfLines(const S: string): Integer;
3827: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3828: procedure DeleteEmptyLines(Ss: TStrings);
3829: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3830:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3831: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3832: {**** files routines - }
3833: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3834:   Resource can be compressed using MS Compress program}
3835: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3836: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3837: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3838: { Execute executes other program and waiting for it terminating, then return its Exit Code }
```

```
3839: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3840: { IniReadSection read section, Section, from ini-file,
3841:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3842:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3843: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3844: { LoadTextFile load text file, FileName, into string }
3845: function LoadTextFile(const FileName: TFileName): string;
3846: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3847: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3848: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3849: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3850: {$IFDEF COMPILER3_UP}
3851: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3852: function TargetFileName(const FileName: TFileName): TFileName;
3853: { return filename ShortCut linked to }
3854: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3855: {$ENDIF COMPILER3_UP}
3856: {**** Graphic routines - }
3857: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3858: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3859: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3860: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3861: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3862: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3863: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3864: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3865: { Cinema draws some visual effect }
3866: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3867: { Roughed fills rect with special 3D pattern }
3868: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3869: { 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 }
3870: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3871: { TextWidth calculate text with for writing using standard desktop font }
3872: function TextWidth(AStr: string): Integer;
3873: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3874: function DefineCursor(Identifer: PChar): TCursor;
3875: {**** other routines - }
3876: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3877: function FindFormByClass(FormClass: TFormClass): TForm;
3878: function FindFormByClassName(FormClassName: string): TForm;
3879: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3880:   having Tag property value, equaled to Tag parameter }
3881: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3882: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3883: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3884: { RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3885: function RBTag(Parent: TWinControl): Integer;
3886: { AppMinimized returns True, if Application is minimized }
3887: function AppMinimized: Boolean;
3888: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3889:   if Caption parameter = '', it replaced with Application.Title }
3890: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
3891: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3892:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3893: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3894:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3895: { Delay stop program execution to MSec msec }
3896: procedure Delay(MSec: Longword);
3897: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3898: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3899: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3900: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3901: function PanelBorder(Panel: TCustomPanel): Integer;
3902: function Pixels(Control: TControl; APixels: Integer): Integer;
3903: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3904: procedure Error(const Msg: string);
3905: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3906:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3907:   {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3908: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3909:   const HideSelColor: Boolean): string;
3910: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3911:   const HideSelColor: Boolean): Integer;
3912: function ItemHtPlain(const Text: string): string;
3913: { ClearList - clears list of TObject }
3914: procedure ClearList(List: TList);
3915: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3916: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3917: { RTTI support }
3918: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3919: function GetPropStr(Obj: TObject; const PropName: string): string;
3920: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3921: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3922: procedure PrepareIniSection(SS: TStrings);
3923: { following functions are not documented because they are don't work properly, so don't use them }
3924: {$IFDEF COMPILER2}
3925: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3926: {$ENDIF}
```

```
3927:
3928: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3929: begin
3930:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3931:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3932:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
3933:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3934:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3935:  Procedure BoxSetItem( List : TWinControl; Index : Integer)
3936:  Function BoxGetFirstSelection( List : TWinControl) : Integer
3937:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3938: end;
3939:
3940: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3941: begin
3942:  Const('MaxInitStrNum','LongInt'( 9);
3943: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
      : array of AnsiString; MaxSplit : Integer) : Integer
3944: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
      string; MaxSplit : Integer) : Integer
3945: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
      QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3946: Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3947: Function JvStrStrip( S : string) : string
3948: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3949: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
3950: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3951: Function StrEatWhiteSpace( const S : string) : string
3952: Function HexToAscii( const S : AnsiString) : AnsiString
3953: Function AsciiToHex( const S : AnsiString) : AnsiString
3954: Function StripQuotes( const S1 : AnsiString) : AnsiString
3955: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3956: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3957: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3958: Function HexPCharToInt( S1 : PAnsiChar) : Integer
3959: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
3960: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
3961: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
3962: Function JvValidIdentifier( S1 : String) : Boolean
3963: Function JvEndChar( X : AnsiChar) : Boolean
3964: Procedure JvGetToken( S1, S2 : PAnsiChar)
3965: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
3966: Function IsKeyword( S1 : PAnsiChar) : Boolean
3967: Function JvValidVarReference( S1 : PAnsiChar) : Boolean
3968: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
3969: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
3970: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
3971: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
3972: Function GetTokenCount : Integer
3973: Procedure ResetTokenCount
3974: end;
3975:
3976: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
3977: begin
3978:   SIRegister_TJvQueryParamsDialog(CL);
3979:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
3980: end;
3981:
3982: ******************************* JvStrUtil /  JvStrUtils;****************************
3983: function FindNotBlankCharPos(const S: string): Integer;
3984: function AnsiChangeCase(const S: string): string;
3985: function GetWordOnPos(const S: string; const P: Integer): string;
3986: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3987: function Cmp(const S1, S2: string): Boolean;
3988: { Spaces returns string consists on N space chars }
3989: function Spaces(const N: Integer): string;
3990: { HasChar returns True, if char, Ch, contains in string, S }
3991: function HasChar(const Ch: Char; const S: string): Boolean;
3992: function HasAnyChar(const Chars: string; const S: string): Boolean;
3993: { SubStr returns substring from string, S, separated with Separator string}
3994: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3995: { SubStrEnd same to previous function but Index numerated from the end of string }
3996: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3997: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
3998: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3999: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
4000: { GetXYByPos is same to previous function, but returns X position in line too}
4001: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4002: { AddSlash returns string with added slash char to Dir parameter, if needed }
4003: function AddSlash2(const Dir: TFileName): string;
4004: { AddPath returns FileName with Path, if FileName not contain any path }
4005: function AddPath(const FileName, Path: TFileName): TFileName;
4006: { ExePath returns ExtractFilePath(ParamStr(0)) }
4007: function ExePath: TFileName;
4008: function LoadTextFile(const FileName: TFileName): string;
4009: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4010: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4011: function ConcatSep(const S, S2, Separator: string): string;
```

```
4012: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4013: function FileEquMask(FileName, Mask: TFileName): Boolean;
4014: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4015:   Masks must be separated with comma (';') }
4016: function FileEquMasks(FileName, Masks: TFileName): Boolean;
4017: function StringEndsWith(const Str, SubStr: string): Boolean;
4018: function ExtractFilePath2(const FileName: string): string;
4019: function StrToOem(const AnsiStr: string): string;
4020: { StrToOem translates a string from the Windows character set into the OEM character set. }
4021: function OemToAnsiStr(const OemStr: string): string;
4022: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4023: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4024: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4025: function ReplaceStr(const S, Srch, Replace: string): string;
4026: { Returns string with every occurrence of Srch string replaced with Replace string. }
4027: function DelSpace(const S: string): string;
4028: { DelSpace return a string with all white spaces removed. }
4029: function DelChars(const S: string; Chr: Char): string;
4030: { DelChars return a string with all Chr characters removed. }
4031: function DelBSpace(const S: string): string;
4032: { DelBSpace trims leading spaces from the given string. }
4033: function DelESpace(const S: string): string;
4034: { DelESpace trims trailing spaces from the given string. }
4035: function DelRSpace(const S: string): string;
4036: { DelRSpace trims leading and trailing spaces from the given string. }
4037: function DelSpace1(const S: string): string;
4038: { DelSpace1 return a string with all non-single white spaces removed. }
4039: function Tab2Space(const S: string; Numb: Byte): string;
4040: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4041: function NPos(const C: string; S: string; N: Integer): Integer;
4042: { NPos searches for a N-th position of substring C in a given string. }
4043: function MakeStr(C: Char; N: Integer): string;
4044: function MS(C: Char; N: Integer): string;
4045: { MakeStr return a string of length N filled with character C. }
4046: function AddChar(C: Char; const S: string; N: Integer): string;
4047: { AddChar return a string left-padded to length N with characters C. }
4048: function AddCharR(C: Char; const S: string; N: Integer): string;
4049: { AddCharR return a string right-padded to length N with characters C. }
4050: function LeftStr(const S: string; N: Integer): string;
4051: { LeftStr return a string right-padded to length N with blanks. }
4052: function RightStr(const S: string; N: Integer): string;
4053: { RightStr return a string left-padded to length N with blanks. }
4054: function CenterStr(const S: string; Len: Integer): string;
4055: { CenterStr centers the characters in the string based upon the Len specified. }
4056: function CompStr(const S1, S2: string): Integer;
4057: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4058: function CompText(const S1, S2: string): Integer;
4059: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4060: function Copy2Symb(const S: string; Symb: Char): string;
4061: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4062: function Copy2SymbDel(var S: string; Symb: Char): string;
4063: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4064: function Copy2Space(const S: string): string;
4065: { Copy2Symb returns a substring of a string S from begining to first white space. }
4066: function Copy2SpaceDel(var S: string): string;
4067: { Copy2SpaceDel returns a substring of a string S from begining to first
4068:   white space and removes this substring from S. }
4069: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4070: { Returns string, with the first letter of each word in uppercase,
4071:   all other letters in lowercase. Words are delimited by WordDelims. }
4072: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4073: { WordCount given a set of word delimiters, returns number of words in S. }
4074: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4075: { Given a set of word delimiters, returns start position of N'th word in S. }
4076: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4077: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4078: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4079: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4080:   delimiters, return the N'th word in S. }
4081: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4082: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4083: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4084: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4085: function QuotedString(const S: string; Quote: Char): string;
4086: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4087: function ExtractQuotedString(const S: string; Quote: Char): string;
4088: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4089:    and reduces pairs of Quote characters within the quoted string to a single character. }
4090: function FindPart(const HelpWilds, InputStr: string): Integer;
4091: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4092: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4093: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4094: function XorString(const Key, Src: ShortString): ShortString;
4095: function XorEncode(const Key, Source: string): string;
4096: function XorDecode(const Key, Source: string): string;
4097: { ** Command line routines ** }
4098: {$IFNDEF COMPILER4_UP}
4099: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4100: {$ENDIF}
```

```
4101: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4102: { ** Numeric string handling routines ** }
4103: function Numb2USA(const S: string): string;
4104: { Numb2USA converts numeric string S to USA-format. }
4105: function Dec2Hex(N: Longint; A: Byte): string;
4106: function D2H(N: Longint; A: Byte): string;
4107: { Dec2Hex converts the given value to a hexadecimal string representation
4108:   with the minimum number of digits (A) specified. }
4109: function Hex2Dec(const S: string): Longint;
4110: function H2D(const S: string): Longint;
4111: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4112: function Dec2Numb(N: Longint; A, B: Byte): string;
4113: { Dec2Numb converts the given value to a string representation with the
4114:   base equal to B and with the minimum number of digits (A) specified. }
4115: function Numb2Dec(S: string; B: Byte): Longint;
4116: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4117: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4118: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4119: function IntToRoman(Value: Longint): string;
4120: { IntToRoman converts the given value to a roman numeric string representation. }
4121: function RomanToInt(const S: string): Longint;
4122: { RomanToInt converts the given string to an integer value. If the string
4123:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4124: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4125: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4126: ********************************* JvFileUtil;*********************************
4127: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4128: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4129: procedure MoveFile(const FileName, DestName: TFileName);
4130: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4131: {$IFDEF COMPILER4_UP}
4132: function GetFileSize(const FileName: string): Int64;
4133: {$ELSE}
4134: function GetFileSize(const FileName: string): Longint;
4135: {$ENDIF}
4136: function FileDateTime(const FileName: string): TDateTime;
4137: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4138: function DeleteFiles(const FileMask: string): Boolean;
4139: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4140: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4141: function NormalDir(const DirName: string): string;
4142: function RemoveBackSlash(const DirName: string): string;
4143: function ValidFileName(const FileName: string): Boolean;
4144: function DirExists(Name: string): Boolean;
4145: procedure ForceDirectories(Dir: string);
4146: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4147:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4148: {$IFDEF COMPILER4_UP}
4149: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4150: {$ENDIF}
4151: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4152: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4153: {$IFDEF COMPILER4_UP}
4154: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4155: {$ENDIF}
4156: function GetTempDir: string;
4157: function GetWindowsDir: string;
4158: function GetSystemDir: string;
4159: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4160: {$IFDEF WIN32}
4161: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4162: function ShortToLongFileName(const ShortName: string): string;
4163: function ShortToLongPath(const ShortName: string): string;
4164: function LongToShortFileName(const LongName: string): string;
4165: function LongToShortPath(const LongName: string): string;
4166: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4167: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4168: {$ENDIF WIN32}
4169: {$IFNDEF COMPILER3_UP}
4170: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4171: {$ENDIF}
4172: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4173: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4174: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4175: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4176:
4177: //*************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4178:  Procedure VariantClear( var V : Variant)
4179:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4180:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4181:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4182:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4183:  Procedure VariantSub( const src : Variant; var dst : Variant)
4184:  Procedure VariantMul( const src : Variant; var dst : Variant)
4185:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4186:  Procedure VariantMod( const src : Variant; var dst : Variant)
4187:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4188:  Procedure VariantOr( const src : Variant; var dst : Variant)
```

```
4189:  Procedure VariantXor( const src : Variant; var dst : Variant)
4190:  Procedure VariantShl( const src : Variant; var dst : Variant)
4191:  Procedure VariantShr( const src : Variant; var dst : Variant)
4192:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4193:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4194:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4195:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4196:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4197:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4198:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4199:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4200:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4201:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4202:  Function VariantNot( const V1 : Variant) : Variant
4203:  Function VariantNeg( const V1 : Variant) : Variant
4204:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4205:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4206:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4207:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4208:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4209:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4210:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4211:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4212:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4213:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4214: end;
4215:
4216: ************************************unit uPSI_JvgUtils;************************************
4217: function IsEven(I: Integer): Boolean;
4218: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4219: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4220: procedure SwapInt(var I1, I2: Integer);
4221: function Spaces(Count: Integer): string;
4222: function DupStr(const Str: string; Count: Integer): string;
4223: function DupChar(C: Char; Count: Integer): string;
4224: procedure Msg(const AMsg: string);
4225: function RectW(R: TRect): Integer;
4226: function RectH(R: TRect): Integer;
4227: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4228: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4229: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4230: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4231:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4232: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4233: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4234:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor,HighlightColor,ShadowColor:
      TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4235: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4236: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4237:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4238: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4239: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4240: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4241:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4242:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4243:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4244: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4245:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4246:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4247:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4248: procedure BringParentWindowToTop(Wnd: TWinControl);
4249: function GetParentForm(Control: TControl): TForm;
4250: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4251: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4252: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4253: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4254: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4255: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4256: function CalcMathString(AExpression: string): Single;
4257: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4258: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4259: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4260: procedure TypeStringOnKeyboard(const S: string);
4261: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4262: procedure DrawTextExtAligned(Canvas:TCanvas;const
      Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4263: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4264: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4265: function ComponentToString(Component: TComponent): string;
4266: procedure StringToComponent(Component: TComponent; const Value: string);
4267: function PlayWaveResource(const ResName: string): Boolean;
4268: function UserName: string;
4269: function ComputerName: string;
4270: function CreateIniFileName: string;
4271: function ExpandString(const Str: string; Len: Integer): string;
4272: function Transliterate(const Str: string; RusToLat: Boolean): string;
4273: function IsSmallFonts: Boolean;
4274: function SystemColorDepth: Integer;
4275: function GetFileTypeJ(const FileName: string): TglFileType;
```

```
4276: Function GetFileType( hFile : THandle) : DWORD');
4277: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4278: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4279:
4280: { ************************************Utility routines of unit classes}
4281: function LineStart(Buffer, BufPos: PChar): PChar
4282: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4283:   'Strings: TStrings): Integer
4284:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4285:  Procedure RegisterClass( AClass : TPersistentClass)
4286:  Procedure RegisterClasses( AClasses : array of TPersistentClass)
4287:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4288:  Procedure UnRegisterClass( AClass : TPersistentClass)
4289:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4290:  Procedure UnRegisterModuleClasses( Module : HMODULE)
4291:  Function FindGlobalComponent( const Name : string) : TComponent
4292:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4293:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4294:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4295:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4296:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4297:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4298:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4299:  Procedure GlobalFixupReferences
4300:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4301:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4302:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4303:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4304:  Procedure RemoveFixups( Instance : TPersistent)
4305:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
4306:  Procedure BeginGlobalLoading
4307:  Procedure NotifyGlobalLoading
4308:  Procedure EndGlobalLoading
4309:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4310:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4311:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4312:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4313:  Procedure FreeObjectInstance( ObjectInstance : Pointer)
4314: // Function AllocateHWnd( Method : TWndMethod) : HWND
4315:  Procedure DeallocateHWnd( Wnd : HWND)
4316:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4317: ************************************unit uPSI_SqlTimSt and DB;************************************
4318: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4319: Function VarSQLTimeStampCreate3: Variant;
4320: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4321: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4322: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4323: Function VarSQLTimeStamp : TVarType
4324: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4325: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4326: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4327: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4328: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4329: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4330: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4331: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4332: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4333: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4334: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4335: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4336: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4337:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4338: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4339: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4340: Procedure DisposeMem( var Buffer, Size : Integer)
4341: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4342: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4343: Function VarTypeToDataType( VarType : Integer) : TFieldType
4344: ************************************************unit JvStrings;************************************
4345: {template functions}
4346: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4347: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4348: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4349: function RemoveMasterBlocks(const SourceStr: string): string;
4350: function RemoveFields(const SourceStr: string): string;
4351: {http functions}
4352: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4353: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4354: {set functions}
4355: procedure SplitSet(AText: string; AList: TStringList);
4356: function JoinSet(AList: TStringList): string;
4357: function FirstOfSet(const AText: string): string;
4358: function LastOfSet(const AText: string): string;
4359: function CountOfSet(const AText: string): Integer;
4360: function SetRotateRight(const AText: string): string;
4361: function SetRotateLeft(const AText: string): string;
4362: function SetPick(const AText: string; AIndex: Integer): string;
4363: function SetSort(const AText: string): string;
4364: function SetUnion(const Set1, Set2: string): string;
```

```
4365: function SetIntersect(const Set1, Set2: string): string;
4366: function SetExclude(const Set1, Set2: string): string;
4367: {replace any <,> etc by &lt; &gt;}
4368: function XMLSafe(const AText: string): string;
4369: {simple hash, Result can be used in Encrypt}
4370: function Hash(const AText: string): Integer;
4371: { Base64 encode and decode a string }
4372: function B64Encode(const S: AnsiString): AnsiString;
4373: function B64Decode(const S: AnsiString): AnsiString;
4374: {Basic encryption from a Borland Example}
4375: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4376: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4377: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4378: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4379: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4380: procedure CSVToTags(Src, Dst: TStringList);
4381: // converts a csv list to a tagged string list
4382: procedure TagsToCSV(Src, Dst: TStringList);
4383: // converts a tagged string list to a csv list
4384: // only fieldnames from the first record are scanned ib the other records
4385: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4386: {selects akey=avalue from Src and returns recordset in Dst}
4387: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4388: {filters Src for akey=avalue}
4389: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4390: {orders a tagged Src list by akey}
4391: function PosStr(const FindString, SourceString: string;
4392:   StartPos: Integer = 1): Integer;
4393: { PosStr searches the first occurrence of a substring FindString in a string
4394:   given by SourceString with case sensitivity (upper and lower case characters
4395:   are differed). This function returns the index value of the first character
4396:   of a specified substring from which it occurs in a given string starting with
4397:   StartPos character index. If a specified substring is not found Q_PosStr
4398:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4399: function PosStrLast(const FindString, SourceString: string): Integer;
4400: {finds the last occurance}
4401: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4402: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4403: { PosText searches the first occurrence of a substring FindString in a string
4404:   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
4405: function PosTextLast(const FindString, SourceString: string): Integer;
4406: {finds the last occurance}
4407: function NameValuesToXML(const AText: string): string;
4408: {$IFDEF MSWINDOWS}
4409: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4410: {$ENDIF MSWINDOWS}
4411: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4412: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4413: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4414: procedure SaveString(const AFile, AText: string);
4415: Procedure SaveStringasFile( const AFile, AText : string)
4416: function LoadStringJ(const AFile: string): string;
4417: Function LoadStringofFile( const AFile : string) : string
4418: Procedure SaveStringtoFile( const AFile, AText : string)
4419: Function LoadStringfromFile( const AFile : string) : string
4420: function HexToColor(const AText: string): TColor;
4421: function UppercaseHTMLTags(const AText: string): string;
4422: function LowercaseHTMLTags(const AText: string): string;
4423: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4424: function RelativePath(const ASrc, ADst: string): string;
4425: function GetToken(var Start: Integer; const SourceText: string): string;
4426: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4427: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4428: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4429: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4430: // parses the beginning of an attribute: space + alpha character
4431: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4432: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4433: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4434: // parses all name=value attributes to the attributes TStringList
4435: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4436: // checks if a name="value" pair exists and returns any value
4437: function GetStrValue(const AText, AName, ADefault: string): string;
4438: // retrieves string value from a line like:
4439: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4440: // returns ADefault when not found
4441: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4442: // same for a color
4443: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4444: // same for an Integer
4445: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4446: // same for a float
4447: function GetBoolValue(const AText, AName: string): Boolean;
4448: // same for Boolean but without default
4449: function GetValue(const AText, AName: string): string;
4450: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4451: procedure SetValue(var AText: string; const AName, AValue: string);
```

```
4452: // sets a string value in a line
4453: procedure DeleteValue(var AText: string; const AName: string);
4454: // deletes a AName="value" pair from AText
4455: procedure GetNames(AText: string; AList: TStringList);
4456: // get a list of names from a string with name="value" pairs
4457: function GetHTMLColor(AColor: TColor): string;
4458: // converts a color value to the HTML hex value
4459: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4460: // finds a string backward case sensitive
4461: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4462: // finds a string backward case insensitive
4463: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4464:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4465: // finds a text range, e.g. <TD>....</TD> case sensitive
4466: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4467:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4468: // finds a text range, e.g. <TD>....</td> case insensitive
4469: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4470:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4471: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4472: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4473:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4474: // finds a text range backward, e.g. <TD>....</td> case insensitive
4475: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4476:   var RangeEnd: Integer): Boolean;
4477: // finds a HTML or XML tag:  <....>
4478: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4479:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4480: // finds the innertext between opening and closing tags
4481: function Easter(NYear: Integer): TDateTime;
4482: // returns the easter date of a year.
4483: function GetWeekNumber(Today: TDateTime): string;
4484: //gets a datecode. Returns year and weeknumber in format: YYWW
4485: function ParseNumber(const S: string): Integer;
4486: // parse number returns the last position, starting from 1
4487: function ParseDate(const S: string): Integer;
4488: // parse a SQL style data string from positions 1,
4489: // starts and ends with #
4490:
4491: ****************************************unit JvJCLUtils;****************************************
4492:
4493: function VarIsInt(Value: Variant): Boolean;
4494:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4495: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4496: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4497: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4498: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4499: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4500: function GetWordOnPos(const S: string; const P: Integer): string;
4501: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4502: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4503: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4504: { GetWordOnPosEx working like GetWordOnPos function, but
4505:   also returns Word position in iBeg, iEnd variables }
4506: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4507: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4508: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4509: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4510: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4511: { GetEndPosCaret returns the caret position of the last char. For the position
4512:   after the last char of Text you must add 1 to the returned X value. }
4513: procedure GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Integer;var X,Y:Integer);
4514: { GetEndPosCaret returns the caret position of the last char. For the position
4515:   after the last char of Text you must add 1 to the returned X value. }
4516: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4517: function SubStrBySeparator(const S:string;const Index:Integer;const
     Separator:string;StartIndex:Int=1):string;
4518: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
     Separator:WideString;StartIndex:Int:WideString;
4519: { SubStrEnd same to previous function but Index numerated from the end of string }
4520: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4521: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4522: function SubWord(P: PChar; var P2: PChar): string;
4523: function CurrencyByWord(Value: Currency): string;
4524: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4525: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4526: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4527: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4528: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4529: { ReplaceString searches for all substrings, OldPattern,
4530:   in a string, S, and replaces them with NewPattern }
4531: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4532: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
     WideString;StartIndex:Integer=1):WideString;
4533: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4534: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4535: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4536: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
```

```
4537:
4538: { Next 4 function for russian chars transliterating.
4539:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4540: procedure Dos2Win(var S: AnsiString);
4541: procedure Win2Dos(var S: AnsiString);
4542: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4543: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4544: function Win2Koi(const S: AnsiString): AnsiString;
4545: { FillString fills the string Buffer with Count Chars }
4546: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4547: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4548: { MoveString copies Count Chars from Source to Dest }
4549: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE} overload;
4550: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4551:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4552: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4553: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4554: { MoveWideChar copies Count WideChars from Source to Dest }
4555: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
      SUPPORTS_INLINE}
4556: { FillNativeChar fills Buffer with Count NativeChars }
4557: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4558: { MoveWideChar copies Count WideChars from Source to Dest }
4559: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4560: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4561: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4562: { Spaces returns string consists on N space chars }
4563: function Spaces(const N: Integer): string;
4564: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4565: function AddSpaces(const S: string; const N: Integer): string;
4566: function SpacesW(const N: Integer): WideString;
4567: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4568: { function LastDateRUS for russian users only }
4569: { returns date relative to current date: 'ääà äíý íàçàä' }
4570: function LastDateRUS(const Dat: TDateTime): string;
4571: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4572: function CurrencyToStr(const Cur: Currency): string;
4573: { HasChar returns True, if Char, Ch, contains in string, S }
4574: function HasChar(const Ch: Char; const S: string): Boolean;
4575: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4576: function HasAnyChar(const Chars: string; const S: string): Boolean;
4577: {$IFNDEF COMPILER12_UP}
4578: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4579: {$ENDIF ~COMPILER12_UP}
4580: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
      SUPPORTS_INLINE}
4581: function CountOfChar(const Ch: Char; const S: string): Integer;
4582: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4583: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4584: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4585: function StrPosW(S, SubStr: PWideChar): PWideChar;
4586: function StrLenW(S: PWideChar): Integer;
4587: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4588: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4589: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4590: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4591: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4592:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4593: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4594: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4595: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4596: Procedure GrayscaleBitmap( Bitmap : TBitmap)
4597: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream
4598: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4599: Function ScreenPixelFormat : TPixelFormat
4600: Function ScreenColorCount : Integer
4601: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4602: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4603: //  SIRegister_TJvGradient(CL);
4604:
4605: {************************************ files routines}
4606: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4607: const
4608:   {$IFDEF MSWINDOWS}
4609:   DefaultCaseSensitivity = False;
4610:   {$ENDIF MSWINDOWS}
4611:   {$IFDEF UNIX}
4612:   DefaultCaseSensitivity = True;
4613:   {$ENDIF UNIX}
4614: { GenTempFileName returns temporary file name on
4615:   drive, there FileName is placed }
4616: function GenTempFileName(FileName: string): string;
4617: { GenTempFileNameExt same to previous function, but
4618:   returning filename has given extension, FileExt }
```

```
4619: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4620: { ClearDir clears folder Dir }
4621: function ClearDir(const Dir: string): Boolean;
4622: { DeleteDir clears and than delete folder Dir }
4623: function DeleteDir(const Dir: string): Boolean;
4624: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4625: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4626: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4627:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4628: function FileEquMasks(FileName, Masks: TFileName;
4629:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4630: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4631: {$IFDEF MSWINDOWS}
4632: { LZFileExpand expand file, FileSource,
4633:   into FileDest. Given file must be compressed, using MS Compress program }
4634: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4635: {$ENDIF MSWINDOWS}
4636: { FileGetInfo fills SearchRec record for specified file attributes}
4637: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4638: { HasSubFolder returns True, if folder APath contains other folders }
4639: function HasSubFolder(APath: TFileName): Boolean;
4640: { IsEmptyFolder returns True, if there are no files or
4641:   folders in given folder, APath}
4642: function IsEmptyFolder(APath: TFileName): Boolean;
4643: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4644: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4645: { AddPath returns FileName with Path, if FileName not contain any path }
4646: function AddPath(const FileName, Path: TFileName): TFileName;
4647: function AddPaths(const PathList, Path: string): string;
4648: function ParentPath(const Path: TFileName): TFileName;
4649: function FindInPath(const FileName, PathList: string): TFileName;
4650: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4651: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4652: { HasParam returns True, if program running with specified parameter, Param }
4653: function HasParam(const Param: string): Boolean;
4654: function HasSwitch(const Param: string): Boolean;
4655: function Switch(const Param: string): string;
4656: { ExePath returns ExtractFilePath(ParamStr(0)) }
4657: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4658: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4659: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4660: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4661: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4662: {**** Graphic routines }
4663: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4664: function IsTTFontSelected(const DC: HDC): Boolean;
4665: function KeyPressed(VK: Integer): Boolean;
4666: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4667: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4668: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4669: {**** Color routines }
4670: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4671: function RGBToBGR(Value: Cardinal): Cardinal;
4672: //function ColorToPrettyName(Value: TColor): string;
4673: //function PrettyNameToColor(const Value: string): TColor;
4674: {**** other routines }
4675: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4676: function IntPower(Base, Exponent: Integer): Integer;
4677: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4678: function StrToBool(const S: string): Boolean;
4679: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4680: function VarToInt(V: Variant): Integer;
4681: function VarToFloat(V: Variant): Double;
4682: { following functions are not documented because they not work properly sometimes, so do not use them }
4683: // (rom) ReplaceStrings1, GetSubStr removed
4684: function GetLongFileName(const FileName: string): string;
4685: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4686: function GetParameter: string;
4687: function GetComputerID: string;
4688: function GetComputerName: string;
4689: {**** string routines }
4690: { ReplaceAllStrings searches for all substrings, Words,
4691:   in a string, S, and replaces them with Frases with the same Index. }
4692: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4693: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4694:   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 }
4695: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4696: { CountOfLines calculates the lines count in a string, S,
4697:   each line must be separated from another with CrLf sequence }
4698: function CountOfLines(const S: string): Integer;
4699: { DeleteLines deletes all lines from strings which in the words,  words.
4700:   The word of will be deleted from strings. }
4701: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4702: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4703:   Lines contained only spaces also deletes. }
4704: procedure DeleteEmptyLines(Ss: TStrings);
4705: { SQLAddWhere addes or modifies existing where-statement, where,
4706:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
```

```
4707:   it must be started on the begining of any line }
4708: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4709: {**** files routines - }
4710: {$IFDEF MSWINDOWS}
4711: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4712:   Resource can be compressed using MS Compress program}
4713: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4714: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
     Boolean;
4715: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4716: {$ENDIF MSWINDOWS}
4717: { IniReadSection read section, Section, from ini-file,
4718:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4719:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4720: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4721: { LoadTextFile load text file, FileName, into string }
4722: function LoadTextFile(const FileName: TFileName): string;
4723: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4724: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4725: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4726: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4727: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4728: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4729: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4730: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4731: { RATextCalcHeight calculate needed height for
4732:   correct output, using RATextOut or RATextOutEx functions }
4733: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4734: { Cinema draws some visual effect }
4735: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4736: { Roughed fills rect with special 3D pattern }
4737: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4738: { 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 }
4739: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4740: { TextWidth calculate text with for writing using standard desktop font }
4741: function TextWidth(const AStr: string): Integer;
4742: { TextHeight calculate text height for writing using standard desktop font }
4743: function TextHeight(const AStr: string): Integer;
4744: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4745: procedure Error(const Msg: string);
4746: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4747:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4748: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4749: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4750:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4751: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4752:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4753: function ItemHtPlain(const Text: string): string;
4754: { ClearList - clears list of TObject }
4755: procedure ClearList(List: TList);
4756: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
4757: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4758: { RTTI support }
4759: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4760: function GetPropStr(Obj: TObject; const PropName: string): string;
4761: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4762: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4763: procedure PrepareIniSection(Ss: TStrings);
4764: { following functions are not documented because they are don't work properly, so don't use them }
4765: // (rom) from JvBandWindows to make it obsolete
4766: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4767: // (rom) from JvBandUtils to make it obsolete
4768: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4769: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4770: function CreateIconFromClipboard: TIcon;
4771: { begin JvIconClipboardUtils } { Icon clipboard routines }
4772: function CF_ICON: Word;
4773: procedure AssignClipboardIcon(Icon: TIcon);
4774: { Real-size icons support routines (32-bit only) }
4775: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4776: function CreateRealSizeIcon(Icon: TIcon): HICON;
4777: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4778: {end JvIconClipboardUtils }
4779: function CreateScreenCompatibleDC: HDC;
4780: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF}
4781: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4782: { begin JvRLE } // (rom) changed API for inclusion in JCL
4783: procedure RleCompressTo(InStream, OutStream: TStream);
4784: procedure RleDecompressTo(InStream, OutStream: TStream);
4785: procedure RleCompress(Stream: TStream);
4786: procedure RleDecompress(Stream: TStream);
4787: { end JvRLE } { begin JvDateUtil }
4788: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4789: function IsLeapYear(AYear: Integer): Boolean;
4790: function DaysInAMonth(const AYear, AMonth: Word): Word;
4791: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4792: function FirstDayOfPrevMonth: TDateTime;
```

```
4793: function LastDayOfPrevMonth: TDateTime;
4794: function FirstDayOfNextMonth: TDateTime;
4795: function ExtractDay(ADate: TDateTime): Word;
4796: function ExtractMonth(ADate: TDateTime): Word;
4797: function ExtractYear(ADate: TDateTime): Word;
4798: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4799: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4800: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4801: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4802: function ValidDate(ADate: TDateTime): Boolean;
4803: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4804: function MonthsBetween(Date1, Date2: TDateTime): Double;
4805: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4806: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4807: function DaysBetween(Date1, Date2: TDateTime): Longint;
4808: { The same as previous but if Date2 < Date1 result = 0 }
4809: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4810: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4811: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4812: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4813: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4814: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4815: { String to date conversions }
4816: function GetDateOrder(const DateFormat: string): TDateOrder;
4817: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4818: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4819: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4820: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4821: //function DefDateFormat(AFourDigitYear: Boolean): string;
4822: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4823: function FormatLongDate(Value: TDateTime): string;
4824: function FormatLongDateTime(Value: TDateTime): string;
4825: { end JvDateUtil }
4826: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4827: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4828: { begin JvStrUtils } { ** Common string handling routines ** }
4829: {$IFDEF UNIX}
4830: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4831:   const ToCode, FromCode: AnsiString): Boolean;
4832: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4833: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4834: function OemStrToAnsi(const S: AnsiString): AnsiString;
4835: function AnsiStrToOem(const S: AnsiString): AnsiString;
4836: {$ENDIF UNIX}
4837: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4838: { StrToOem translates a string from the Windows character set into the OEM character set. }
4839: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4840: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4841: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4842: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4843: function ReplaceStr(const S, Srch, Replace: string): string;
4844: { Returns string with every occurrence of Srch string replaced with Replace string. }
4845: function DelSpace(const S: string): string;
4846: { DelSpace return a string with all white spaces removed. }
4847: function DelChars(const S: string; Chr: Char): string;
4848: { DelChars return a string with all Chr characters removed. }
4849: function DelBSpace(const S: string): string;
4850: { DelBSpace trims leading spaces from the given string. }
4851: function DelESpace(const S: string): string;
4852: { DelESpace trims trailing spaces from the given string. }
4853: function DelRSpace(const S: string): string;
4854: { DelRSpace trims leading and trailing spaces from the given string. }
4855: function DelSpace1(const S: string): string;
4856: { DelSpace1 return a string with all non-single white spaces removed. }
4857: function Tab2Space(const S: string; Numb: Byte): string;
4858: { Tab2Space converts any tabulation character in the given string to the
4859:   Numb spaces characters. }
4860: function NPos(const C: string; S: string; N: Integer): Integer;
4861: { NPos searches for a N-th position of substring C in a given string. }
4862: function MakeStr(C: Char; N: Integer): string; overload;
4863: {$IFNDEF COMPILER12_UP}
4864: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4865: {$ENDIF !COMPILER12_UP}
4866: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4867: { MakeStr return a string of length N filled with character C. }
4868: function AddChar(C: Char; const S: string; N: Integer): string;
4869: { AddChar return a string left-padded to length N with characters C. }
4870: function AddCharR(C: Char; const S: string; N: Integer): string;
4871: { AddCharR return a string right-padded to length N with characters C. }
4872: function LeftStr(const S: string; N: Integer): string;
4873: { LeftStr return a string right-padded to length N with blanks. }
4874: function RightStr(const S: string; N: Integer): string;
4875: { RightStr return a string left-padded to length N with blanks. }
4876: function CenterStr(const S: string; Len: Integer): string;
4877: { CenterStr centers the characters in the string based upon the Len specified. }
4878: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4879: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4880:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4881: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
```

```
4882: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4883: function Copy2Symb(const S: string; Symb: Char): string;
4884: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4885: function Copy2SymbDel(var S: string; Symb: Char): string;
4886: { Copy2SymbDel returns a substring of a string S from begining to first
4887:   character Symb and removes this substring from S. }
4888: function Copy2Space(const S: string): string;
4889: { Copy2Symb returns a substring of a string S from begining to first white space. }
4890: function Copy2SpaceDel(var S: string): string;
4891: { Copy2SpaceDel returns a substring of a string S from begining to first
4892:   white space and removes this substring from S. }
4893: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4894: { Returns string, with the first letter of each word in uppercase,
4895:   all other letters in lowercase. Words are delimited by WordDelims. }
4896: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4897: { WordCount given a set of word delimiters, returns number of words in S. }
4898: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4899: { Given a set of word delimiters, returns start position of N'th word in S. }
4900: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4901: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4902: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4903: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4904:   delimiters, return the N'th word in S. }
4905: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4906: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4907:   that started from position Pos. }
4908: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4909: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4910: function QuotedString(const S: string; Quote: Char): string;
4911: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4912: function ExtractQuotedString(const S: string; Quote: Char): string;
4913: { ExtractQuotedString removes the Quote characters from the beginning and
4914:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4915: function FindPart(const HelpWilds, InputStr: string): Integer;
4916: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4917: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4918: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4919: function XorString(const Key, Src: ShortString): ShortString;
4920: function XorEncode(const Key, Source: string): string;
4921: function XorDecode(const Key, Source: string): string;
4922: { ** Command line routines ** }
4923: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4924: { ** Numeric string handling routines ** }
4925: function Numb2USA(const S: string): string;
4926: { Numb2USA converts numeric string S to USA-format. }
4927: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4928: { Dec2Hex converts the given value to a hexadecimal string representation
4929:   with the minimum number of digits (A) specified. }
4930: function Hex2Dec(const S: string): Longint;
4931: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4932: function Dec2Numb(N: Int64; A, B: Byte): string;
4933: { Dec2Numb converts the given value to a string representation with the
4934:   base equal to B and with the minimum number of digits (A) specified. }
4935: function Numb2Dec(S: string; B: Byte): Int64;
4936: { Numb2Dec converts the given B-based numeric string to the corresponding
4937:   integer value. }
4938: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4939: { IntToBin converts the given value to a binary string representation
4940:   with the minimum number of digits specified. }
4941: function IntToRoman(Value: Longint): string;
4942: { IntToRoman converts the given value to a roman numeric string representation. }
4943: function RomanToInt(const S: string): Longint;
4944: { RomanToInt converts the given string to an integer value. If the string
4945:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4946: function FindNotBlankCharPos(const S: string): Integer;
4947: function FindNotBlankCharPosW(const S: WideString): Integer;
4948: function AnsiChangeCase(const S: string): string;
4949: function WideChangeCase(const S: string): string;
4950: function StartsText(const SubStr, S: string): Boolean;
4951: function EndsText(const SubStr, S: string): Boolean;
4952: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4953: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4954: {end JvStrUtils}
4955: {$IFDEF UNIX}
4956: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4957: {$ENDIF UNIX}
4958: { begin JvFileUtil }
4959: function FileDateTime(const FileName: string): TDateTime;
4960: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4961: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4962: function NormalDir(const DirName: string): string;
4963: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4964: function ValidFileName(const FileName: string): Boolean;
4965: {$IFDEF MSWINDOWS}
4966: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4967: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4968: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4969: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4970: {$ENDIF MSWINDOWS}
```

```
4971: function GetWindowsDir: string;
4972: function GetSystemDir: string;
4973: function ShortToLongFileName(const ShortName: string): string;
4974: function LongToShortFileName(const LongName: string): string;
4975: function ShortToLongPath(const ShortName: string): string;
4976: function LongToShortPath(const LongName: string): string;
4977: {$IFDEF MSWINDOWS}
4978: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4979: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4980: {$ENDIF MSWINDOWS}
4981: { end JvFileUtil }
4982: // Works like PtInRect but includes all edges in comparision
4983: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4984: // Works like PtInRect but excludes all edges from comparision
4985: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4986: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4987: function IsFourDigitYear: Boolean;
4988: { moved from JvJVCLUTils }
4989: //Open an object with the shell (url or something like that)
4990: function OpenObject(const Value: string): Boolean; overload;
4991: function OpenObject(Value: PChar): Boolean; overload;
4992: {$IFDEF MSWINDOWS}
4993: //Raise the last Exception
4994: procedure RaiseLastWin32; overload;
4995: procedure RaiseLastWin32(const Text: string); overload;
4996: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
4997: function GetFileVersion(const AFileName: string): Cardinal;
4998: {$EXTERNALSYM GetFileVersion}
4999: //Get version of Shell.dll
5000: function GetShellVersion: Cardinal;
5001: {$EXTERNALSYM GetShellVersion}
5002: // CD functions  on HW
5003: procedure OpenCdDrive;
5004: procedure CloseCdDrive;
5005: // returns True if Drive is accessible
5006: function DiskInDrive(Drive: Char): Boolean;
5007: {$ENDIF MSWINDOWS}
5008: //Same as linux function ;)
5009: procedure PError(const Text: string);
5010: // execute a program without waiting
5011: procedure Exec(const FileName, Parameters, Directory: string);
5012: // execute a program and wait for it to finish
5013: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5014: // returns True if this is the first instance of the program that is running
5015: function FirstInstance(const ATitle: string): Boolean;
5016: // restores a window based on it's classname and Caption. Either can be left empty
5017: // to widen the search
5018: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5019: // manipulate the traybar and start button
5020: procedure HideTraybar;
5021: procedure ShowTraybar;
5022: procedure ShowStartButton(Visible: Boolean = True);
5023: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5024: procedure MonitorOn;
5025: procedure MonitorOff;
5026: procedure LowPower;
5027: // send a key to the window named AppName
5028: function SendKey(const AppName: string; Key: Char): Boolean;
5029: {$IFDEF MSWINDOWS}
5030: // returns a list of all win currently visible, the Objects property is filled with their window handle
5031: procedure GetVisibleWindows(List: TStrings);
5032: Function GetVisibleWindowsF( List : TStrings):TStrings');
5033: // associates an extension to a specific program
5034: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5035: procedure AddToRecentDocs(const FileName: string);
5036: function GetRecentDocs: TStringList;
5037: {$ENDIF MSWINDOWS}
5038: function CharIsMoney(const Ch: Char): Boolean;
5039: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5040: function IntToExtended(I: Integer): Extended;
5041: { GetChangedText works out the new text given the current cursor pos & the key pressed
5042:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5043: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5044: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5045: //function StrIsInteger(const S: string): Boolean;
5046: function StrIsFloatMoney(const Ps: string): Boolean;
5047: function StrIsDateTime(const Ps: string): Boolean;
5048: function PreformatDateString(Ps: string): string;
5049: function BooleanToInteger(const B: Boolean): Integer;
5050: function StringToBoolean(const Ps: string): Boolean;
5051: function SafeStrToDateTime(const Ps: string): TDateTime;
5052: function SafeStrToDate(const Ps: string): TDateTime;
5053: function SafeStrToTime(const Ps: string): TDateTime;
5054: function StrDelete(const psSub, psMain: string): string;
5055:   { returns the fractional value of pcValue}
5056: function TimeOnly(pcValue: TDateTime): TTime;
5057: { returns the integral value of pcValue }
```

```
5058: function DateOnly(pcValue: TDateTime): TDate;
5059: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5060: const { TDateTime value used to signify Null value}
5061:   NullEquivalentDate: TDateTime = 0.0;
5062: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5063: // Replacement for Win32Check to avoid platform specific warnings in D6
5064: function OSCheck(RetVal: Boolean): Boolean;
5065: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5066:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5067:   not be forced to use FileCtrl unnecessarily }
5068: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5069: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5070: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5071: function MinimizeString(const S: string; const MaxLen: Integer): string;
5072: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
      SW_SHOWDEFAULT);
5073: { 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.}
5074: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5075: {$ENDIF MSWINDOWS}
5076: procedure ResourceNotFound(ResID: PChar);
5077: function EmptyRect: TRect;
5078: function RectWidth(R: TRect): Integer;
5079: function RectHeight(R: TRect): Integer;
5080: function CompareRect(const R1, R2: TRect): Boolean;
5081: procedure RectNormalize(var R: TRect);
5082: function RectIsSquare(const R: TRect): Boolean;
5083: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5084: //If AMaxSize = -1 ,then auto calc Square's max size
5085: {$IFDEF MSWINDOWS}
5086: procedure FreeUnusedOle;
5087: function GetWindowsVersion: string;
5088: function LoadDLL(const LibName: string): THandle;
5089: function RegisterServer(const ModuleName: string): Boolean;
5090: function UnregisterServer(const ModuleName: string): Boolean;
5091: {$ENDIF MSWINDOWS}
5092: { String routines }
5093: function GetEnvVar(const VarName: string): string;
5094: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5095: function StringToPChar(var S: string): PChar;
5096: function StrPAlloc(const S: string): PChar;
5097: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5098: function DropT(const S: string): string;
5099: { Memory routines }
5100: function AllocMemo(Size: Longint): Pointer;
5101: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5102: procedure FreeMemo(var fpBlock: Pointer);
5103: function GetMemoSize(fpBlock: Pointer): Longint;
5104: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5105: { Manipulate huge pointers routines }
5106: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5107: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5108: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5109: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5110: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5111: function WindowClassName(Wnd: THandle): string;
5112: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5113: procedure ActivateWindow(Wnd: THandle);
5114: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5115: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5116: { SetWindowTop put window to top without recreating window }
5117: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5118: procedure CenterWindow(Wnd: THandle);
5119: function MakeVariant(const Values: array of Variant): Variant;
5120: { Convert dialog units to pixels and backwards }
5121: {$IFDEF MSWINDOWS}
5122: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5123: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5124: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5125: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5126: {$ENDIF MSWINDOWS}
5127: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5128: {$IFDEF BCB}
5129: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5130: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5131: {$ELSE}
5132: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5133: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5134: {$ENDIF BCB}
5135: {$IFDEF MSWINDOWS}
5136: { BrowseForFolderNative displays Browse For Folder dialog }
5137: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5138: {$ENDIF MSWINDOWS}
5139: procedure AntiAlias(Clip: TBitmap);
5140: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5141: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5142:   ABitmap: TBitmap; const SourceRect: TRect);
5143: function IsTrueType(const FontName: string): Boolean;
```

```
5144: // Removes all non-numeric characters from AValue and returns the resulting string
5145: function TextToValText(const AValue: string): string;
5146: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5147: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5148: Function ReplaceRegExpr(const ARegExpr,AInputStr,
      AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5149: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5150: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5151:
5152: ****************************************************unit uPSI_JvTFUtils;
5153:  Function JExtractYear( ADate : TDateTime) : Word
5154:  Function JExtractMonth( ADate : TDateTime) : Word
5155:  Function JExtractDay( ADate : TDateTime) : Word
5156:  Function ExtractHours( ATime : TDateTime) : Word
5157:  Function ExtractMins( ATime : TDateTime) : Word
5158:  Function ExtractSecs( ATime : TDateTime) : Word
5159:  Function ExtractMSecs( ATime : TDateTime) : Word
5160:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5161:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5162:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5163:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5164:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5165:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5166:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5167:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5168:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5169:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5170:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5171:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
5172:  Procedure EnsureMonth( Month : Word)
5173:  Procedure EnsureDOW( DOW : Word)
5174:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5175:  Function Lesser( N1, N2 : Integer) : Integer
5176:  Function Greater( N1, N2 : Integer) : Integer
5177:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5178:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5179:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5180:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5181:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5182:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5183:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
      AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5184:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
      HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5185:  Function JRectWidth( ARect : TRect) : Integer
5186:  Function JRectHeight( ARect : TRect) : Integer
5187:  Function JEmptyRect : TRect
5188:  Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5189:
5190: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5191: begin
5192:  Procedure HideTaskBarButton( hWindow : HWND)
5193:  Function msLoadStr( ID : Integer) : String
5194:  Function msFormat( fmt : String; params : array of const) : String
5195:  Function msFileExists( const FileName : String) : Boolean
5196:  Function msIntToStr( Int : Int64) : String
5197:  Function msStrPas( const Str : PChar) : String
5198:  Function msRenameFile( const OldName, NewName : String) : Boolean
5199:  Function CutFileName( s : String) : String
5200:  Function GetVersionInfo( var VersionString : String) : DWORD
5201:  Function FormatTime( t : Cardinal) : String
5202:  Function msCreateDir( const Dir : string) : Boolean
5203:  Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5204:  Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5205:  Function msStrLen( Str : PChar) : Integer
5206:  Function msDirectoryExists( const Directory : String) : Boolean
5207:  Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5208:  Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5209:  Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5210:  Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5211:  Function GetTextFromFile( Filename : String) : string
5212:  Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5213:  Function msStrToIntDef( const s : String; const i : Integer) : Integer
5214:  Function msStrToInt( s : String) : Integer
5215:  Function GetItemText( hDlg : THandle; ID : DWORD) : String
5216: end;
5217:
5218: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5219: begin
5220:   //TDynFloatArray', 'array of Extended
5221:   TDynLWordArray', 'array of LongWord
5222:   TDynLIntArray', 'array of LongInt
5223:   TDynFloatMatrix', 'array of TDynFloatArray
5224:   TDynLWordMatrix', 'array of TDynLWordArray
5225:   TDynLIntMatrix', 'array of TDynLIntArray
5226:  Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5227:  Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5228:  Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5229:  Function Log10All( const X : TDynFloatArray) : TDynFloatArray
```

```
5230:  Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5231:  Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5232:  Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5233:  Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5234:  Function DotProduct( const X, Y : TDynFloatArray) : Extended
5235:  Function MNorm( const X : TDynFloatArray) : Extended
5236:  Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5237:  Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5238:  Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5239:  Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5240:  Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5241:  Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5242:  Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5243:  Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5244:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5245:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5246:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5247:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5248:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5249: end;
5250:
5251: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5252: begin
5253:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5254:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5255:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5256:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5257:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5258:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5259:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5260:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5261:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5262:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5263:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5264:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5265:  'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5266:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5267:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5268:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5269:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5270:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5271:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5272:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5273:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5274:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5275:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5276:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5277:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5278:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5279:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5280:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5281:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5282:  'ESBLn10','Extended').setExtended( 2.3025850929940456840 2);
5283:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5284:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5285:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5286:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5287:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5288:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5289:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5290:  'ESBInvPi','Extended').setExtended( 3.18309886183790671 54e-1);
5291:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5292:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5293:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5294:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5295:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5296:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5297:  'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5298:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5299:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5300:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5301:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5302:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5303:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5304:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5305:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5306:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5307:   //LongWord', 'Cardinal
5308:   TBitList', 'Word
5309:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5310:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5311:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5312:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5313:  Function SameFloat( const X1, X2 : Extended) : Boolean
5314:  Function FloatIsZero( const X : Extended) : Boolean
5315:  Function FloatIsPositive( const X : Extended) : Boolean
5316:  Function FloatIsNegative( const X : Extended) : Boolean
5317:  Procedure IncLim( var B : Byte; const Limit : Byte)
5318:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
```

```
5319:  Procedure IncLimW( var B : Word; const Limit : Word)
5320:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5321:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5322:  Procedure DecLim( var B : Byte; const Limit : Byte)
5323:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5324:  Procedure DecLimW( var B : Word; const Limit : Word)
5325:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5326:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5327:  Function MaxB( const B1, B2 : Byte) : Byte
5328:  Function MinB( const B1, B2 : Byte) : Byte
5329:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5330:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5331:  Function MaxW( const B1, B2 : Word) : Word
5332:  Function MinW( const B1, B2 : Word) : Word
5333:  Function esbMaxI( const B1, B2 : Integer) : Integer
5334:  Function esbMinI( const B1, B2 : Integer) : Integer
5335:  Function MaxL( const B1, B2 : LongInt) : LongInt
5336:  Function MinL( const B1, B2 : LongInt) : LongInt
5337:  Procedure SwapB( var B1, B2 : Byte)
5338:  Procedure SwapSI( var B1, B2 : ShortInt)
5339:  Procedure SwapW( var B1, B2 : Word)
5340:  Procedure SwapI( var B1, B2 : SmallInt)
5341:  Procedure SwapL( var B1, B2 : LongInt)
5342:  Procedure SwapI32( var B1, B2 : Integer)
5343:  Procedure SwapC( var B1, B2 : LongWord)
5344:  Procedure SwapInt64( var X, Y : Int64)
5345:  Function esbSign( const B : LongInt) : ShortInt
5346:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5347:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5348:  Function Max3Word( const X1, X2, X3 : Word) : Word
5349:  Function Min3Word( const X1, X2, X3 : Word) : Word
5350:  Function MaxBArray( const B : array of Byte) : Byte
5351:  Function MaxWArray( const B : array of Word) : Word
5352:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5353:  Function MaxIArray( const B : array of Integer) : Integer
5354:  Function MaxLArray( const B : array of LongInt) : LongInt
5355:  Function MinBArray( const B : array of Byte) : Byte
5356:  Function MinWArray( const B : array of Word) : Word
5357:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5358:  Function MinIArray( const B : array of Integer) : Integer
5359:  Function MinLArray( const B : array of LongInt) : LongInt
5360:  Function SumBArray( const B : array of Byte) : Byte
5361:  Function SumBArray2( const B : array of Byte) : Word
5362:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5363:  Function SumSIArray2( const B : array of ShortInt) : Integer
5364:  Function SumWArray( const B : array of Word) : Word
5365:  Function SumWArray2( const B : array of Word) : LongInt
5366:  Function SumIArray( const B : array of Integer) : Integer
5367:  Function SumLArray( const B : array of LongInt) : LongInt
5368:  Function SumLWArray( const B : array of LongWord) : LongWord
5369:  Function ESBDigits( const X : LongWord) : Byte
5370:  Function BitsHighest( const X : LongWord) : Integer
5371:  Function ESBBitsNeeded( const X : LongWord) : Integer
5372:  Function esbGCD( const X, Y : LongWord) : LongWord
5373:  Function esbLCM( const X, Y : LongInt) : Int64
5374:  //Function esbLCM( const X, Y : LongInt) : LongInt
5375:  Function RelativePrime( const X, Y : LongWord) : Boolean
5376:  Function Get87ControlWord : TBitList
5377:  Procedure Set87ControlWord( const CWord : TBitList)
5378:  Procedure SwapExt( var X, Y : Extended)
5379:  Procedure SwapDbl( var X, Y : Double)
5380:  Procedure SwapSing( var X, Y : Single)
5381:  Function esbSgn( const X : Extended) : ShortInt
5382:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5383:  Function ExtMod( const X, Y : Extended) : Extended
5384:  Function ExtRem( const X, Y : Extended) : Extended
5385:  Function CompMOD( const X, Y : Comp) : Comp
5386:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5387:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5388:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5389:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5390:  Function MaxExt( const X, Y : Extended) : Extended
5391:  Function MinExt( const X, Y : Extended) : Extended
5392:  Function MaxEArray( const B : array of Extended) : Extended
5393:  Function MinEArray( const B : array of Extended) : Extended
5394:  Function MaxSArray( const B : array of Single) : Single
5395:  Function MinSArray( const B : array of Single) : Single
5396:  Function MaxCArray( const B : array of Comp) : Comp
5397:  Function MinCArray( const B : array of Comp) : Comp
5398:  Function SumSArray( const B : array of Single) : Single
5399:  Function SumEArray( const B : array of Extended) : Extended
5400:  Function SumSqEArray( const B : array of Extended) : Extended
5401:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5402:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5403:  Function SumCArray( const B : array of Comp) : Comp
5404:  Function FactorialX( A : LongWord) : Extended
5405:  Function PermutationX( N, R : LongWord) : Extended
5406:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5407:  Function IsPositiveEArray( const X : array of Extended) : Boolean
```

```
5408:  Function esbGeometricMean( const X : array of Extended) : Extended
5409:  Function esbHarmonicMean( const X : array of Extended) : Extended
5410:  Function ESBMean( const X : array of Extended) : Extended
5411:  Function esbSampleVariance( const X : array of Extended) : Extended
5412:  Function esbPopulationVariance( const X : array of Extended) : Extended
5413:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5414:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5415:  Function GetMedian( const SortedX : array of Extended) : Extended
5416:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5417:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5418:  Function ESBMagnitude( const X : Extended) : Integer
5419:  Function ESBTan( Angle : Extended) : Extended
5420:  Function ESBCot( Angle : Extended) : Extended
5421:  Function ESBCosec( const Angle : Extended) : Extended
5422:  Function ESBSec( const Angle : Extended) : Extended
5423:  Function ESBArcTan( X, Y : Extended) : Extended
5424:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5425:  Function ESBArcCos( const X : Extended) : Extended
5426:  Function ESBArcSin( const X : Extended) : Extended
5427:  Function ESBArcSec( const X : Extended) : Extended
5428:  Function ESBArcCosec( const X : Extended) : Extended
5429:  Function ESBLog10( const X : Extended) : Extended
5430:  Function ESBLog2( const X : Extended) : Extended
5431:  Function ESBLogBase( const X, Base : Extended) : Extended
5432:  Function Pow2( const X : Extended) : Extended
5433:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5434:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5435:  Function XtoY( const X, Y : Extended) : Extended
5436:  Function esbTenToY( const Y : Extended) : Extended
5437:  Function esbTwoToY( const Y : Extended) : Extended
5438:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5439:  Function esbISqrt( const I : LongWord) : Longword
5440:  Function ILog2( const I : LongWord) : LongWord
5441:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5442:  Function ESBArCosh( X : Extended) : Extended
5443:  Function ESBArSinh( X : Extended) : Extended
5444:  Function ESBArTanh( X : Extended) : Extended
5445:  Function ESBCosh( X : Extended) : Extended
5446:  Function ESBSinh( X : Extended) : Extended
5447:  Function ESBTanh( X : Extended) : Extended
5448:  Function InverseGamma( const X : Extended) : Extended
5449:  Function esbGamma( const X : Extended) : Extended
5450:  Function esbLnGamma( const X : Extended) : Extended
5451:  Function esbBeta( const X, Y : Extended) : Extended
5452:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5453: end;
5454:
5455: ********************************Integer Huge Cardinal Utils**************************
5456: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5457: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5458: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5459: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5460: Function BitCount_8( Value : byte) : integer
5461: Function BitCount_16( Value : uint16) : integer
5462: Function BitCount_32( Value : uint32) : integer
5463: Function BitCount_64( Value : uint64) : integer
5464: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5465:  Procedure ( CountPrimalityTests : integer)
5466:  Function gcd( a, b : THugeCardinal) : THugeCardinal
5467:  Function lcm( a, b : THugeCardinal) : THugeCardinal
5468:  Function isCoPrime( a, b : THugeCardinal) : boolean
5469:  Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5470:  Function hasSmallFactor( p : THugeCardinal) : boolean
5471:  //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
      NumbersTested: integer) : boolean
5472:  Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5473:  Const('StandardExponent','LongInt'( 65537);
5474: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5475:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5476:
5477: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5478: begin
5479:   AddTypeS('TXRTLInteger', 'array of Integer
5480:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5481:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5482:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5483:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5484:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5485:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5486:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5487:   'BitsPerByte','LongInt'( 8);
5488:   BitsPerDigit','LongInt'( 32);
5489:   SignBitMask','LongWord( $80000000);
5490:  Function XRTLAdjustBits( const ABits : Integer) : Integer
5491:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5492:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
```

```
5493:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5494:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5495:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5496:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5497:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5498:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5499:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5500:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5501:  Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5502:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5503:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5504:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5505:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5506:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5507:  Procedure XRTLZero( var AInteger : TXRTLInteger)
5508:  Procedure XRTLOne( var AInteger : TXRTLInteger)
5509:  Procedure XRTLMOne( var AInteger : TXRTLInteger)
5510:  Procedure XRTLTwo( var AInteger : TXRTLInteger)
5511:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5512:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5513:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5514:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5515:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5516:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5517:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5518:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5519:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5520:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5521:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5522:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5523:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5524:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5525:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5526:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5527:  Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger)
5528:  Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5529:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5530:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5531:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5532:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5533:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5534:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5535:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5536:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5537:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5538:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5539:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5540:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5541:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5542:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5543:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5544:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5545:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5546:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5547:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5548:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5549:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5550:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5551:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5552:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5553:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5554:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5555:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5556:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5557:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5558:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5559:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5560:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5561:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5562:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5563:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5564: end;
5565:
5566: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5567: begin
5568:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5569:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5570:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5571:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5572:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const
      AColor:TColor;Rect:TRect;
5573:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5574:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5575:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
```

```
5576:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5577:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5578:  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)
5579: end;
5580:
5581:
5582: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5583: begin
5584:  Function StrDec( S : String) : String
5585:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5586:  Function ReadNumFromEdit( Edit : TEdit) : Float
5587:  Procedure WriteNumToFile( var F : Text; X : Float)
5588: end;
5589:
5590: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5591: begin
5592:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5593:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5594:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5595:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5596:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5597:  Procedure TeX_SetGraphTitle( Title : String)
5598:  Procedure TeX_SetOxTitle( Title : String)
5599:  Procedure TeX_SetOyTitle( Title : String)
5600:  Procedure TeX_PlotOxAxis
5601:  Procedure TeX_PlotOyAxis
5602:  Procedure TeX_PlotGrid( Grid : TGrid)
5603:  Procedure TeX_WriteGraphTitle
5604:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5605:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5606:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5607:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5608:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5609:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5610:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5611:  Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5612:  Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5613:  Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5614:  Function Xcm( X : Float) : Float
5615:  Function Ycm( Y : Float) : Float
5616: end;
5617:
5618: *---------------------------------------------------------------------------*)
5619: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5620: begin
5621:   TConstArray', 'array of TVarRec
5622:  Function CopyVarRec( const Item : TVarRec) : TVarRec
5623:  Function CreateConstArray( const Elements : array of const) : TConstArray
5624:  Procedure FinalizeVarRec( var Item : TVarRec)
5625:  Procedure FinalizeConstArray( var Arr : TConstArray)
5626: end;
5627:
5628: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5629: begin
5630:  Function HexBS( B : Byte) : ShortString
5631:  Function HexWS( W : Word) : ShortString
5632:  Function HexLS( L : LongInt) : ShortString
5633:  Function HexPtrS( P : Pointer) : ShortString
5634:  Function BinaryBS( B : Byte) : ShortString
5635:  Function BinaryWS( W : Word) : ShortString
5636:  Function BinaryLS( L : LongInt) : ShortString
5637:  Function OctalBS( B : Byte) : ShortString
5638:  Function OctalWS( W : Word) : ShortString
5639:  Function OctalLS( L : LongInt) : ShortString
5640:  Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5641:  Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5642:  Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5643:  Function Str2RealS( const S : ShortString; var R : Double) : Boolean
5644:  Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5645:  Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5646:  Function Long2StrS( L : LongInt) : ShortString
5647:  Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5648:  Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5649:  Function ValPrepS( const S : ShortString) : ShortString
5650:  Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5651:  Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5652:  Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5653:  Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5654:  Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5655:  Function TrimLeadS( const S : ShortString) : ShortString
5656:  Function TrimTrailS( const S : ShortString) : ShortString
5657:  Function TrimS( const S : ShortString) : ShortString
5658:  Function TrimSpacesS( const S : ShortString) : ShortString
5659:  Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5660:  Function CenterS( const S : ShortString; Len : Cardinal) : ShortString
5661:  Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5662:  Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
```

```
5663:  Function ScrambleS( const S, Key : ShortString) : ShortString
5664:  Function SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5665:  Function FilterS( const S, Filters : ShortString) : ShortString
5666:  Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5667:  Function CharCountS( const S : ShortString; C : AnsiChar) : Byte
5668:  Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5669:  Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5670:  Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5671:  Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5672:  Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5673:  Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5674:  Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
      Margin:Cardinal;PadToMargin:Boolean)
5675:  Function CompStringS( const S1, S2 : ShortString) : Integer
5676:  Function CompUCStringS( const S1, S2 : ShortString) : Integer
5677:  Function SoundexS( const S : ShortString) : ShortString
5678:  Function MakeLetterSetS( const S : ShortString) : Longint
5679:  Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5680:  Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
      Pos:Cardinal):Bool;
5681:  Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
      Pos:Cardinal):Bool;
5682:  Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5683:  Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5684:  Function JustFilenameS( const PathName : ShortString) : ShortString
5685:  Function JustNameS( const PathName : ShortString) : ShortString
5686:  Function JustExtensionS( const Name : ShortString) : ShortString
5687:  Function JustPathnameS( const PathName : ShortString) : ShortString
5688:  Function AddBackSlashS( const DirName : ShortString) : ShortString
5689:  Function CleanPathNameS( const PathName : ShortString) : ShortString
5690:  Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5691:  Function CommaizeS( L : LongInt) : ShortString
5692:  Function CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5693:  Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
      DecPt:Char):ShortString;
5694:  Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5695:  Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5696:  Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5697:  Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5698:  Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5699:  Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5700:  Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5701:  Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5702:  Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5703:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5704:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5705:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5706:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5707:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5708:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
      SubString:ShortString):Bool;
5709:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
      SubStr:ShortString):Bool;
5710:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5711:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5712:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5713:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5714:  Function ExtractTokensS(const S,
      Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5715:  Function IsChAlphaS( C : Char) : Boolean
5716:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5717:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5718:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5719:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5720:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5721:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5722:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5723:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5724:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5725:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5726:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
      Cardinal):ShortString;
5727:  Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
      Replacements:Cardinal):ShortString;
5728:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5729:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
      Replacements:Cardinal):ShortString
5730:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
      Replacements:Cardinal):ShortString
5731:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5732:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5733:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5734:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5735: end;
5736:
5737:
```

```
5738: ********unit uPSI_StUtils; from Systools4*************************************************************
5739: Function SignL( L : LongInt) : Integer
5740: Function SignF( F : Extended) : Integer
5741: Function MinWord( A, B : Word) : Word
5742: Function MidWord( W1, W2, W3 : Word) : Word
5743: Function MaxWord( A, B : Word) : Word
5744: Function MinLong( A, B : LongInt) : LongInt
5745: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5746: Function MaxLong( A, B : LongInt) : LongInt
5747: Function MinFloat( F1, F2 : Extended) : Extended
5748: Function MidFloat( F1, F2, F3 : Extended) : Extended
5749: Function MaxFloat( F1, F2 : Extended) : Extended
5750: Function MakeInteger16( H, L : Byte) : SmallInt
5751: Function MakeWordS( H, L : Byte) : Word
5752: Function SwapNibble( B : Byte) : Byte
5753: Function SwapWord( L : LongInt) : LongInt
5754: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5755: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5756: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5757: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5758: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5759: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5760: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5761: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5762: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5763: Procedure ExchangeBytes( var I, J : Byte)
5764: Procedure ExchangeWords( var I, J : Word)
5765: Procedure ExchangeLongInts( var I, J : LongInt)
5766: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5767: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5768: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5769: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5770: //*****************uPSI_StFIN;************************************************************
5771: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5772: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
      Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5773: Function BondDuration( Settlement,Maturity:TStDate;Rate,
      Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5774: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
      Extended
5775: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5776: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5777: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5778: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5779: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5780: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5781: Function DollarToDecimalText( DecDollar : Extended) : string
5782: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5783: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5784: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5785: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5786: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5787: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5788: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5789: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5790: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5791: Function IsCardValid( const S : string) : Boolean
5792: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5793: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5794: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5795: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5796: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5797: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5798: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5799: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5800: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended
5801: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5802: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended
5803: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5804: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5805: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5806: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5807: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5808: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5809: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5810: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5811: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5812:
5813: //****************************************unit uPSI_StAstroP;
5814: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
```

```
5815: //*****unit unit uPSI_StStat; Statistic Package of SysTools*****************************************
5816: Function AveDev( const Data : array of Double) : Double
5817: Function AveDev16( const Data, NData : Integer) : Double
5818: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5819: Function Correlation( const Data1, Data2 : array of Double) : Double
5820: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5821: Function Covariance( const Data1, Data2 : array of Double) : Double
5822: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5823: Function DevSq( const Data : array of Double) : Double
5824: Function DevSq16( const Data, NData : Integer) : Double
5825: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5826:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5827: Function GeometricMeanS( const Data : array of Double) : Double
5828: Function GeometricMean16( const Data, NData : Integer) : Double
5829: Function HarmonicMeanS( const Data : array of Double) : Double
5830: Function HarmonicMean16( const Data, NData : Integer) : Double
5831: Function Largest( const Data : array of Double; K : Integer) : Double
5832: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5833: Function MedianS( const Data : array of Double) : Double
5834: Function Median16( const Data, NData : Integer) : Double
5835: Function Mode( const Data : array of Double) : Double
5836: Function Mode16( const Data, NData : Integer) : Double
5837: Function Percentile( const Data : array of Double; K : Double) : Double
5838: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5839: Function PercentRank( const Data : array of Double; X : Double) : Double
5840: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5841: Function Permutations( Number, NumberChosen : Integer) : Extended
5842: Function Combinations( Number, NumberChosen : Integer) : Extended
5843: Function FactorialS( N : Integer) : Extended
5844: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5845: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5846: Function Smallest( const Data : array of Double; K : Integer) : Double
5847: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5848: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5849: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5850:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5851:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5852: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5853: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5854: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5855: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5856: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5857: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5858: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5859: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5860: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5861: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5862: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5863: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5864: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5865: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5866: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5867: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5868: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5869: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5870: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5871: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5872: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5873: Function NormSDist( Z : Single) : Single
5874: Function NormSInv( Probability : Single) : Single
5875: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5876: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5877: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5878: Function Erfc( X : Single) : Single
5879: Function GammaLn( X : Single) : Single
5880: Function LargestSort( const Data : array of Double; K : Integer) : Double
5881: Function SmallestSort( const Data : array of double; K : Integer) : Double
5882:
5883: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5884:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5885:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5886:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5887:  Function DefaultMergeName( MergeNum : Integer) : string
5888:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5889:
5890: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5891: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5892:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5893:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5894:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5895:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5896:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5897:  Function LunarPhase( UT : TStDateTimeRec) : Double
5898:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5899:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5900:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5901:  Function FullMoon( D : TStDate) : TStLunarRecord
```

```
5902:  Function LastQuarter( D : TStDate) : TStLunarRecord
5903:  Function NewMoon( D : TStDate) : TStLunarRecord
5904:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5905:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5906:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5907:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5908:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5909:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5910:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5911:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5912:  Function SiderealTime( UT : TStDateTimeRec) : Double
5913:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5914:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5915:  Function SEaster( Y, Epoch : Integer) : TStDate
5916:  Function DateTimeToAJD( D : TDateTime) : Double
5917:  Function HoursMin( RA : Double) : ShortString
5918:  Function DegsMin( DC : Double) : ShortString
5919:  Function AJDToDateTime( D : Double) : TDateTime
5920:
5921: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5922:  Function CurrentDate : TStDate
5923:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5924:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5925:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5926:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5927:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5928:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5929:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5930:  Function WeekOfYear( Julian : TStDate) : Byte
5931:  Function AstJulianDate( Julian : TStDate) : Double
5932:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5933:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5934:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5935:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5936:  Function StIsLeapYear( Year : Integer) : Boolean
5937:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5938:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5939:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5940:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5941:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5942:  Function CurrentTime : TStTime
5943:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5944:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5945:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5946:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5947:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5948:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5949:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5950:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5951:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5952:  Function StDateToDateTime( D : TStDate) : TDateTime
5953:  Function StTimeToDateTime( T : TStTime) : TDateTime
5954:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5955:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5956:
5957: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5958: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5959:  Function MonthToString( const Month : Integer) : string
5960:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5961:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5962:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5963:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
5964:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5965:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5966:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5967:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5968:  Function TimeStringToStTime( const Picture, S : string) : TStTime
5969:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5970:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5971:  Function DateStringIsBlank( const Picture, S : string) : Boolean
5972:  Function InternationalDate( ForceCentury : Boolean) : string
5973:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
5974:  Function InternationalTime( ShowSeconds : Boolean) : string
5975:  Procedure ResetInternationalInfo
5976:
5977: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5978:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
5979:  Function AnsiUpperCaseShort32( const S : string) : string
5980:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
5981:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
5982:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
5983:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5984:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
5985:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
5986:  Function Upcase( C : AnsiChar) : AnsiChar
5987:  Function LoCase( C : AnsiChar) : AnsiChar
5988:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
5989:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
5990:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
```

```
5991:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5992:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5993:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
5994:  Procedure RaiseContainerError( Code : longint)
5995:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
5996:  Function ProductOverflow( A, B : LongInt) : Boolean
5997:  Function StNewStr( S : string) : PShortString
5998:  Procedure StDisposeStr( PS : PShortString)
5999:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
6000:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
6001:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
6002:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6003:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6004:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
6005:
6006: procedure SIRegister_usvd(CL: TPSPascalCompiler);
6007: begin
6008:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
6009:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
6010:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
6011:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
6012:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6013: end;
6014:
6015: //**********unit unit ; StMath Package of SysTools*****************************************
6016: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
6017: Function PowerS( Base, Exponent : Extended) : Extended
6018: Function StInvCos( X : Double) : Double
6019: Function StInvSin( Y : Double) : Double
6020: Function StInvTan2( X, Y : Double) : Double
6021: Function StTan( A : Double) : Double
6022: Procedure DumpException;  //unit StExpEng;
6023: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6024:
6025: //**********unit unit ; StCRC Package of SysTools*****************************************
6026: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6027: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6028: Function Adler32OfFile( FileName : AnsiString) : LongInt
6029: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6030: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6031: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6032: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6033: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6034: Function Crc32OfFile( FileName : AnsiString) : LongInt
6035: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6036: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6037: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6038: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6039: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6040: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6041:
6042: //**********unit unit ; StBCD Package of SysTools*****************************************
6043: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6044: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6045: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6046: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6047: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6048: Function NegBcd( const B : TbcdS) : TbcdS
6049: Function AbsBcd( const B : TbcdS) : TbcdS
6050: Function FracBcd( const B : TbcdS) : TbcdS
6051: Function IntBcd( const B : TbcdS) : TbcdS
6052: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6053: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6054: Function ValBcd( const S : string) : TbcdS
6055: Function LongBcd( L : LongInt) : TbcdS
6056: Function ExtBcd( E : Extended) : TbcdS
6057: Function ExpBcd( const B : TbcdS) : TbcdS
6058: Function LnBcd( const B : TbcdS) : TbcdS
6059: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6060: Function PowBcd( const B, E : TbcdS) : TbcdS
6061: Function SqrtBcd( const B : TbcdS) : TbcdS
6062: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6063: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6064: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6065: Function IsIntBcd( const B : TbcdS) : Boolean
6066: Function TruncBcd( const B : TbcdS) : LongInt
6067: Function BcdExt( const B : TbcdS) : Extended
6068: Function RoundBcd( const B : TbcdS) : LongInt
6069: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6070: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6071: Function FormatBcd( const Format : string; const B : TbcdS) : string
6072: Function StrGeneralBcd( const B : TbcdS) : string
6073: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6074: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6075:
6076: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools*********************
6077: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6078: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6079: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
```

```
6080: Function StDeEscape( const EscStr : AnsiString) : Char
6081: Function StDoEscape( Delim : Char) : AnsiString
6082: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6083: Function AnsiHashText( const S : string; Size : Integer) : Integer
6084: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6085: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6086: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6087:
6088: //**********unit unit ; StNetCon Package of SysTools*****************************************
6089:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6090:     Constructor Create( AOwner : TComponent)
6091:     Function Connect : DWord
6092:     Function Disconnect : DWord
6093:     RegisterProperty('Password', 'String', iptrw);
6094:     Property('UserName', 'String', iptrw);
6095:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6096:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6097:     Property('LocalDevice', 'String', iptrw);
6098:     Property('ServerName', 'String', iptrw);
6099:     Property('ShareName', 'String', iptrw);
6100:     Property('OnConnect', 'TNotifyEvent', iptrw);
6101:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6102:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6103:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6104:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6105:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6106:   end;
6107: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6108: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6109: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6110: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6111: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6112: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6113: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6114: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6115: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6116: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6117: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6118: Function SuspendThread( hThread : THandle) : DWORD
6119: Function ResumeThread( hThread : THandle) : DWORD
6120: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6121: Function GetCurrentThread : THandle
6122: Procedure ExitThread( dwExitCode : DWORD)
6123: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6124: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6125: Procedure EndThread(ExitCode: Integer);
6126: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6127: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6128: Procedure FreeProcInstance( Proc : FARPROC)
6129: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6130: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6131: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6132: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6133: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6134: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6135: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6136: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6137: Function CurrentParallelJobInfo : TParallelJobInfo
6138: Function ObtainParallelJobInfo : TParallelJobInfo
6139: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6140: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL');
6141: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL');
6142: Function
      DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
      TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL');
6143: Function SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL');
6144: Function DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
      lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL; dwOptions : DWORD) : BOOL');
6145: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL');
6146: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL');
6147:
6148: ************************************************unit uPSI_JclMime;
6149:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6150:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6151:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6152:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6153:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6154:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6155:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6156:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6157:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6158:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6159:
6160: ************************************************unit uPSI_JclPrint;
6161:  Procedure DirectPrint( const Printer, Data : string)
6162:  Procedure SetPrinterPixelsPerInch
```

```
6163:  Function GetPrinterResolution : TPoint
6164:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6165:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6166:
6167:
6168: //*************************************unit uPSI_ShLwApi;************************************
6169:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6170:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6171:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6172:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6173:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6174:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6175:  Function StrDup( lpSrch : PChar) : PChar
6176:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6177:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6178:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6179:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6180: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6181:  Function StrPBrk( psz, pszSet : PChar) : PChar
6182:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6183:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6184:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6185:  Function StrSpn( psz, pszSet : PChar) : Integer
6186:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6187:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6188:  Function StrToInt( lpSrch : PChar) : Integer
6189:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6190:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6191:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6192:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6193:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6194:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6195:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6196:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6197:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6198:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6199:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6200:  SZ_CONTENTTYPE_HTMLA','String 'text/html
6201:  SZ_CONTENTTYPE_HTMLW','String 'text/html
6202:  SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
6203:  SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6204:  SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6205:  SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
6206:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6207:  STIF_DEFAULT','LongWord( $00000000);
6208:  STIF_SUPPORT_HEX','LongWord( $00000001);
6209:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6210:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6211:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6212:  Function PathAddBackslash( pszPath : PChar) : PChar
6213:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6214:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6215:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6216:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6217:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6218:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6219:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6220:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6221:  Function PathFileExists( pszPath : PChar) : BOOL
6222:  Function PathFindExtension( pszPath : PChar) : PChar
6223:  Function PathFindFileName( pszPath : PChar) : PChar
6224:  Function PathFindNextComponent( pszPath : PChar) : PChar
6225:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6226:  Function PathGetArgs( pszPath : PChar) : PChar
6227:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6228:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6229:  Function PathGetCharType( ch : Char) : UINT
6230:  GCT_INVALID','LongWord( $0000);
6231:  GCT_LFNCHAR','LongWord( $0001);
6232:  GCT_SHORTCHAR','LongWord( $0002);
6233:  GCT_WILD','LongWord( $0004);
6234:  GCT_SEPARATOR','LongWord( $0008);
6235:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6236:  Function PathIsDirectory( pszPath : PChar) : BOOL
6237:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6238:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6239:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6240:  Function PathIsRelative( pszPath : PChar) : BOOL
6241:  Function PathIsRoot( pszPath : PChar) : BOOL
6242:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6243:  Function PathIsUNC( pszPath : PChar) : BOOL
6244:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6245:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6246:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6247:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6248:  Function PathIsURL( pszPath : PChar) : BOOL
6249:  Function PathMakePretty( pszPath : PChar) : BOOL
6250:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6251:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
```

```
6252:  Procedure PathQuoteSpaces( lpsz : PChar)
6253:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6254:  Procedure PathRemoveArgs( pszPath : PChar)
6255:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6256:  Procedure PathRemoveBlanks( pszPath : PChar)
6257:  Procedure PathRemoveExtension( pszPath : PChar)
6258:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6259:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6260:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6261:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6262:  Function PathSkipRoot( pszPath : PChar) : PChar
6263:  Procedure PathStripPath( pszPath : PChar)
6264:  Function PathStripToRoot( pszPath : PChar) : BOOL
6265:  Procedure PathUnquoteSpaces( lpsz : PChar)
6266:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6267:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6268:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6269:  Procedure PathUndecorate( pszPath : PChar)
6270:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6271:  URL_SCHEME_INVALID','LongInt'( - 1);
6272:  URL_SCHEME_UNKNOWN','LongInt'( 0);
6273:  URL_SCHEME_FTP','LongInt'( 1);
6274:  URL_SCHEME_HTTP','LongInt'( 2);
6275:  URL_SCHEME_GOPHER','LongInt'( 3);
6276:  URL_SCHEME_MAILTO','LongInt'( 4);
6277:  URL_SCHEME_NEWS','LongInt'( 5);
6278:  URL_SCHEME_NNTP','LongInt'( 6);
6279:  URL_SCHEME_TELNET','LongInt'( 7);
6280:  URL_SCHEME_WAIS','LongInt'( 8);
6281:  URL_SCHEME_FILE','LongInt'( 9);
6282:  URL_SCHEME_MK','LongInt'( 10);
6283:  URL_SCHEME_HTTPS','LongInt'( 11);
6284:  URL_SCHEME_SHELL','LongInt'( 12);
6285:  URL_SCHEME_SNEWS','LongInt'( 13);
6286:  URL_SCHEME_LOCAL','LongInt'( 14);
6287:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6288:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6289:  URL_SCHEME_ABOUT','LongInt'( 17);
6290:  URL_SCHEME_RES','LongInt'( 18);
6291:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6292:  URL_SCHEME', 'Integer
6293:  URL_PART_NONE','LongInt'( 0);
6294:  URL_PART_SCHEME','LongInt'( 1);
6295:  URL_PART_HOSTNAME','LongInt'( 2);
6296:  URL_PART_USERNAME','LongInt'( 3);
6297:  URL_PART_PASSWORD','LongInt'( 4);
6298:  URL_PART_PORT','LongInt'( 5);
6299:  URL_PART_QUERY','LongInt'( 6);
6300:  URL_PART', 'DWORD
6301:  URLIS_URL','LongInt'( 0);
6302:  URLIS_OPAQUE','LongInt'( 1);
6303:  URLIS_NOHISTORY','LongInt'( 2);
6304:  URLIS_FILEURL','LongInt'( 3);
6305:  URLIS_APPLIABLE','LongInt'( 4);
6306:  URLIS_DIRECTORY','LongInt'( 5);
6307:  URLIS_HASQUERY','LongInt'( 6);
6308:  TUrlIs', 'DWORD
6309:  URL_UNESCAPE','LongWord( $10000000);
6310:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6311:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6312:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6313:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6314:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6315:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6316:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6317:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6318:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6319:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6320:  URL_INTERNAL_PATH','LongWord( $00800000);
6321:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6322:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6323:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6324:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6325:  URL_APPLY_DEFAULT','LongWord( $00000001);
6326:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6327:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6328:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6329:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6330:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6331:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6332:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6333:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6334:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6335:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6336:  Function UrlGetLocation( psz1 : PChar) : PChar
6337:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6338:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6339:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6340:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
```

```
6341:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6342:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6343:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6344:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6345:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6346:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6347:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6348:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6349:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6350:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6351:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
      pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6352:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6353:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6354:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6355:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6356:  SHREGDEL_DEFAULT','LongWord( $00000000);
6357:  SHREGDEL_HKCU','LongWord( $00000001);
6358:  SHREGDEL_HKLM','LongWord( $00000010);
6359:  SHREGDEL_BOTH','LongWord( $00000011);
6360:  SHREGENUM_DEFAULT','LongWord( $00000000);
6361:  SHREGENUM_HKCU','LongWord( $00000001);
6362:  SHREGENUM_HKLM','LongWord( $00000010);
6363:  SHREGENUM_BOTH','LongWord( $00000011);
6364:  SHREGSET_HKCU','LongWord( $00000001);
6365:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6366:  SHREGSET_HKLM','LongWord( $00000004);
6367:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6368:  TSHRegDelFlags', 'DWORD
6369:  TSHRegEnumFlags', 'DWORD
6370:  HUSKEY', 'THandle
6371:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6372:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6373:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6374:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6375:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6376:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6377:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6378:  ASSOCF_VERIFY','LongWord( $00000040);
6379:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6380:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6381:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6382:  ASSOCF', 'DWORD
6383:  ASSOCSTR_COMMAND','LongInt'( 1);
6384:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
6385:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
6386:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6387:  ASSOCSTR_NOOPEN','LongInt'( 5);
6388:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6389:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
6390:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6391:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
6392:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6393:  ASSOCSTR_INFOTIP','LongInt'( 11);
6394:  ASSOCSTR_MAX','LongInt'( 12);
6395:  ASSOCSTR', 'DWORD
6396:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6397:  ASSOCKEY_APP','LongInt'( 2);
6398:  ASSOCKEY_CLASS','LongInt'( 3);
6399:  ASSOCKEY_BASECLASS','LongInt'( 4);
6400:  ASSOCKEY_MAX','LongInt'( 5);
6401:  ASSOCKEY', 'DWORD
6402:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6403:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6404:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6405:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6406:  ASSOCDATA_MAX','LongInt'( 5);
6407:  ASSOCDATA', 'DWORD
6408:  ASSOCENUM_NONE','LongInt'( 0);
6409:  ASSOCENUM', 'DWORD
6410:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6411:  SHACF_DEFAULT $00000000);
6412:  SHACF_FILESYSTEM','LongWord( $00000001);
6413:  SHACF_URLHISTORY','LongWord( $00000002);
6414:  SHACF_URLMRU','LongWord( $00000004);
6415:  SHACF_USETAB','LongWord( $00000008);
6416:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6417:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6418:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6419:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6420:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6421:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6422:  Procedure SHSetThreadRef( punk : IUnknown)
6423:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6424:  CTF_INSIST','LongWord( $00000001);
6425:  CTF_THREAD_REF','LongWord( $00000002);
6426:  CTF_PROCESS_REF','LongWord( $00000004);
6427:  CTF_COINIT','LongWord( $00000008);
6428:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
```

```
6429:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6430:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6431:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6432:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6433:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6434:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6435:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6436:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6437:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6438:  Function SetRectEmpty( var lprc : TRect) : BOOL
6439:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6440:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6441:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6442:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6443:
6444:  Function InitializeFlatSB( hWnd : HWND) : Bool
6445:  Procedure UninitializeFlatSB( hWnd : HWND)
6446:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6447:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6448:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6449:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6450:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6451:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6452:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6453:
6454:
6455: //  ****************************************** 204 unit uPSI_ShellAPI;
6456: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6457: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6458: Procedure DragFinish( Drop : HDROP)
6459: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6460: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6461: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6462: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6463: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6464: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6465: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6466: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6467: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6468: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6469: Procedure SHFreeNameMappings( hNameMappings : THandle)
6470:
6471:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6472:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6473:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6474:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6475:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6476:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6477:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6478:  Function SimpleXMLEncode( const S : string) : string
6479:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6480:  Function XMLEncode( const S : string) : string
6481:  Function XMLDecode( const S : string) : string
6482:  Function EntityEncode( const S : string) : string
6483:  Function EntityDecode( const S : string) : string
6484:
6485: procedure RIRegister_CPort_Routines(S: TPSExec);
6486:  Procedure EnumComPorts( Ports : TStrings)
6487:  Procedure ListComPorts( Ports : TStrings)
6488:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6489:  Function GetComPorts: TStringlist;
6490:  Function StrToBaudRate( Str : string) : TBaudRate
6491:  Function StrToStopBits( Str : string) : TStopBits
6492:  Function StrToDataBits( Str : string) : TDataBits
6493:  Function StrToParity( Str : string) : TParityBits
6494:  Function StrToFlowControl( Str : string) : TFlowControl
6495:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6496:  Function StopBitsToStr( StopBits : TStopBits) : string
6497:  Function DataBitsToStr( DataBits : TDataBits) : string
6498:  Function ParityToStr( Parity : TParityBits) : string
6499:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6500:  Function ComErrorsToStr( Errors : TComErrors) : String
6501:
6502:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6503:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6504:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6505:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6506:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6507:  Function GetMessagePos : DWORD
6508:  Function GetMessageTime : Longint
6509:  Function GetMessageExtraInfo : Longint
6510:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6511:  Procedure JAddToRecentDocs( const Filename : string)
6512:  Procedure ClearRecentDocs
6513:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6514:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6515:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6516:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6517:  Function RecycleFile( FileToRecycle : string) : Boolean
```

```
6518:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6519:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6520:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6521:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
       DWORD; var pcbBytesNeeded : DWORD) : BOOL
6522:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
       DWORD; var pcbBytesNeeded : DWORD) : BOOL
6523:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
       DWORD; var pcbBytesNeeded : DWORD) : BOOL
6524:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
       dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
       lpResumeHandle:DWORD;pszGroupName: LP
6525:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
       dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
       lpResumeHandle:DWORD; pszGroupNam
6526:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
       dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
       lpResumeHandle:DWORD; pszGroupName
6527:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6528:
6529: ****************************************** unit uPSI_JclPeImage;
6530:
6531:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6532: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6533:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6534:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
       DWORD) : TJclRebaseImageInfo
6535:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6536:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6537:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6538:  Function PeDoesExportFunction(const FileName:TFileName;const
       FuncName:string;Options:TJclSmartCompOptions):Bool;
6539:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
       ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6540:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
       FunctionName:string;Options:TJclSmartCompOptions):Bool
6541:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
       : string; Options : TJclSmartCompOptions) : Boolean
6542:  Function PeDoesImportLibrary(const FileName:TFileName;const
       LibraryName:string;Recursive:Boolean):Boolean);
6543:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
       Boolean; FullPathName : Boolean) : Boolean
6544:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
       LibraryName:string; IncludeLibNames : Boolean): Boolean
6545:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6546:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6547:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6548:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
       NamesList:TStrings):Bool
6549:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6550:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
       Descript:Bool):Bool;
6551:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6552:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6553:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6554:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6555:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6556:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6557:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
       PImageSectionHeader
6558:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6559:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6560:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
       ___Pointer;
6561:    SIRegister_TJclPeSectionStream(CL);
6562:    SIRegister_TJclPeMapImgHookItem(CL);
6563:    SIRegister_TJclPeMapImgHooks(CL);
6564:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
       NtHeaders:TImageNtHeaders):Boolean
6565:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6566:    Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6567:    TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6568:    TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6569:    TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6570:    TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6571:    TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6572:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
       TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6573:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
       Descript:TJclBorUmDescription):TJclBorUmResult;
6574:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6575:  Function PeBorUnmangleName3( const Name : string) : string;
6576:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6577:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6578:
6579:
6580: //****************** SysTools uPSI_StSystem; ************************************
6581:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6582:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
```

```
6583:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6584:  //Procedure EnumerateDirectories(const
       StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6585:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
       IncludeItem:TIncludeItemFunc);
6586:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6587:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6588:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6589:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6590:  Function FlushOsBuffers( Handle : Integer) : Boolean
6591:  Function GetCurrentUser : AnsiString
6592:  Function GetDiskClass( Drive : Char) : DiskClass
6593:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
       SectorsPerCluster:Cardinal):Bool;
6594:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
       DiskSize:Double):Bool;
6595:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
       DiskSize:Comp):Boolean;
6596:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6597:  Function getDiskSpace2(const path: String; index: integer): int64;
6598:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6599:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6600:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6601:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6602:  Function GetLongPath( const APath : AnsiString) : AnsiString
6603:  Function GetMachineName : AnsiString
6604:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6605:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6606:  Function GetShortPath( const APath : AnsiString) : AnsiString
6607:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6608:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6609:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6610:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6611:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6612:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6613:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6614:  Function IsDriveReady( Drive : Char) : Boolean
6615:  Function IsFile( const FileName : AnsiString) : Boolean
6616:  Function IsFileArchive( const S : AnsiString) : Integer
6617:  Function IsFileHidden( const S : AnsiString) : Integer
6618:  Function IsFileReadOnly( const S : AnsiString) : Integer
6619:  Function IsFileSystem( const S : AnsiString) : Integer
6620:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6621:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6622:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6623:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6624:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6625:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6626:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6627:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6628:  Function ValidDrive( Drive : Char) : Boolean
6629:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6630:
6631: //****************************unit uPSI_JclLANMan;*****************************************
6632: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6633:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6634:  Function DeleteAccount( const Servername, Username : string) : Boolean
6635:  Function DeleteLocalAccount( Username : string) : Boolean
6636:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6637:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6638:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6639:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6640:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6641:  Function LocalGroupExists( const Group : string) : Boolean
6642:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6643:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6644:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6645:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6646:  Function IsLocalAccount( const AccountName : string) : Boolean
6647:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6648:  Function GetRandomString( NumChar : cardinal) : string
6649:
6650: //****************************unit uPSI_cUtils;*****************************************
6651: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6652: Function cIsWinNT : boolean
6653:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
       Multitasking:Boolean;
6654:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6655:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
       CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6656:  Function cGetShortName( FileName : string) : string
6657:  Procedure cShowError( Msg : String)
6658:  Function cCommaStrToStr( s : string; formatstr : string) : string
6659:  Function cIncludeQuoteIfSpaces( s : string) : string
6660:  Function cIncludeQuoteIfNeeded( s : string) : string
6661:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6662:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
```

```
6663:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6664:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6665:  Function cCodeInstoStr( s : string) : string
6666:  Function cStrtoCodeIns( s : string) : string
6667:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6668:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6669:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6670:  Function cPointtoStr( const pt : TPoint) : string
6671:  Function cListtoStr( const List : TStrings) : string
6672:  Function ListtoStr( const List : TStrings) : string
6673:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6674:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6675:  Function cGetFileTyp( const FileName : string) : TUnitType
6676:  Function cGetExTyp( const FileName : string) : TExUnitType
6677:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6678:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6679:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6680:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6681:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6682:  Function cGenMakePath( FileName : String) : String;
6683:  Function cGenMakePath2( FileName : String) : String
6684:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6685:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6686:  Function cCalcMod( Count : Integer) : Integer
6687:  Function cGetVersionString( FileName : string) : string
6688:  Function cCheckChangeDir( var Dir : string) : boolean
6689:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6690:  Function cIsNumeric( s : string) : boolean
6691:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6692:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6693:  Function GetFileTyp( const FileName : string) : TUnitType
6694:  Function Atoi(const aStr: string): integer
6695:  Function Itoa(const aint: integer): string
6696:  Function Atof(const aStr: string): double');
6697:  Function Atol(const aStr: string): longint');
6698:
6699:
6700: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6701: begin
6702:   FindClass('TOBJECT'),'EHTTP
6703:   FindClass('TOBJECT'),'EHTTPParser
6704:   //AnsiCharSet', 'set of AnsiChar
6705:   AnsiStringArray', 'array of AnsiString
6706:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6707:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6708:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6709:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6710:    +'CustomMinVersion : Integer; end
6711:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6712:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6713:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6714:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6715:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6716:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6717:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6718:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6719:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6720:    +'nection, hntOrigin, hntKeepAlive )
6721:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6722:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6723:    +' AnsiString; end
6724:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6725:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6726:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6727:   //PHTTPContentLength', '^THTTPContentLength // will not work
6728:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6729:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6730:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6731:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6732:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6733:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6734:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6735:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6736:    +' CustomStr : AnsiString; end
6737:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6738:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6739:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6740:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6741:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6742:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6743:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6744:    +'m; Custom : AnsiString; end
6745:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6746:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6747:    +' Custom : AnsiString; end
6748:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6749:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6750:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6751:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
```

```
6752:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6753:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6754:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6755:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6756:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6757:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6758:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6759:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6760:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6761:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6762:    +'FieldEnum; List : array of THTTPContentEncoding; end
6763:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6764:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6765:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6766:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6767:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6768:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6769:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6770:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6771:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6772:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6773:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6774:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6775:    +'CustomFieldArray; Custom : AnsiString; end
6776:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6777:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6778:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6779:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6780:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6781:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6782:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6783:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6784:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6785:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6786:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6787:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6788:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6789:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6790:   THTTPFixedHeaders','array[0..42] of AnsiString
6791:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6792:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6793:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6794:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6795:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6796:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6797:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6798:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6799:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6800:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6801:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6802:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6803:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6804:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6805:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6806:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6807:    +' THTTPDateField; Age : THTTPAgeField; end
6808:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6809:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6810:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6811:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6812:  Procedure InitHTTPRequest( var A : THTTPRequest)
6813:  Procedure InitHTTPResponse( var A : THTTPResponse)
6814:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6815:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6816:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6817:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6818:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6819:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6820:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6821:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6822:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6823:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6824:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6825:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6826:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6827:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6828:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6829:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6830:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6831:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6832:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6833:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6834:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6835:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6836:   THTTPStringOption', '( hsoNone )
6837:   THTTPStringOptions', 'set of THTTPStringOption
6838:   FindClass('TOBJECT'),'TAnsiStringBuilder
6839:
6840:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
```

```
6841:  Procedure BuildStrHTTPContentLengthValue(const
       A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6842:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
       B:TAnsiStringBuilder;P:THTTPStringOptions)
6843:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
       P:THTTPStringOptions)
6844:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
       P:THTTPStringOptions)
6845:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
       B : TAnsiStringBuilder; const P : THTTPStringOptions)
6846:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
       THTTPStringOptions)
6847:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
       P:THTTPStringOptions);
6848:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
       TAnsiStringBuilder; const P : THTTPStringOptions)
6849:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6850:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6851:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6852:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6853:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
       P:THTTPStringOptions);
6854:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6855:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
       B:TAnsiStringBuilder;const P:THTTPStringOptions)
6856:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6857:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
       : THTTPStringOptions)
6858:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions)
6859:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
       : THTTPStringOptions)
6860:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6861:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
       : THTTPStringOptions)
6862:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions);
6863:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
       THTTPStringOptions)
6864:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6865:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
       P:THTTPStringOptions);
6866:  Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
       THTTPStringOptions)
6867:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
       TAnsiStringBuilder; const P : THTTPStringOptions)
6868:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
       THTTPStrOptions);
6869:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions);
6870:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
       P:THTTPStringOptions);
6871:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6872:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6873:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6874:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6875:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6876:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6877:  Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
       Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6878:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6879:   SIRegister_THTTPParser(CL);
6880:   FindClass('TOBJECT'),'THTTPContentDecoder
6881:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6882:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6883:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6884:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
6885:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6886:   SIRegister_THTTPContentDecoder(CL);
6887:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6888:   FindClass('TOBJECT'),'THTTPContentReader
6889:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6890:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
       LogLevel:Int;
6891:   SIRegister_THTTPContentReader(CL);
6892:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
6893:   FindClass('TOBJECT'),'THTTPContentWriter
6894:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6895:   SIRegister_THTTPContentWriter(CL);
6896:  Procedure SelfTestcHTTPUtils
6897: end;
```

```
6898:
6899: (*----------------------------------------------------------------------------*)
6900: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6901: begin
6902:  'TLSLibraryVersion','String '1.00
6903:  'TLSError_None','LongInt'( 0);
6904:  'TLSError_InvalidBuffer','LongInt'( 1);
6905:  'TLSError_InvalidParameter','LongInt'( 2);
6906:  'TLSError_InvalidCertificate','LongInt'( 3);
6907:  'TLSError_InvalidState','LongInt'( 4);
6908:  'TLSError_DecodeError','LongInt'( 5);
6909:  'TLSError_BadProtocol','LongInt'( 6);
6910:  Function TLSErrorMessage( const TLSError : Integer) : String
6911:   SIRegister_ETLSError(CL);
6912:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6913:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6914:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6915:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6916:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6917:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6918:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6919:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6920:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6921:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6922:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6923:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6924:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6925:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6926:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6927:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6928:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6929:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6930:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6931:   PTLSRandom', '^TTLSRandom // will not work
6932:  Procedure InitTLSRandom( var Random : TTLSRandom)
6933:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6934:  'TLSSessionIDMaxLen','LongInt'( 32);
6935:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6936:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6937:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6938:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6939:    +'; Signature : TTLSSignatureAlgorithm; end
6940:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6941:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6942:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6943:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6944:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6945:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6946:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6947:    +'nteger; Supported : Boolean; end
6948:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6949:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
6950:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6951:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6952:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6953:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6954:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6955:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6956:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6957:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6958:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
6959:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
      Size:Integer):AnsiString
6960:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6961:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6962:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
      ClientRandom : AnsiString; const Size : Integer) : AnsiString
6963:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6964:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6965:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6966:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:AnsiString) : AnsiString
6967:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6968:    +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6969:    +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6970:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
      IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6971:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
      ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
6972:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
6973:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
6974:  Procedure SelfTestcTLSUtils
6975: end;
6976:
6977: (*----------------------------------------------------------------------------*)
6978: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
```

```
6979: begin
6980:   sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
      integer; disks : integer; mx : integer; my : integer; end
6981:  // pBoard', '^tBoard // will not work
6982:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6983:  Function rCheckMove( color : byte; cx, cy : integer) : integer
6984:  //Function rDoStep( data : pBoard) : word
6985:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6986: end;
6987:
6988: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
6989: begin
6990: Function InEditMode( ADataset : TDataset) : Boolean
6991: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
6992: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6993: Function GetFieldText( AField : TField) : String
6994: end;
6995:
6996: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
6997: begin
6998:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6999:   TMyPrintRange', '( prAll, prSelected )
7000:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7001:    +'ded, ssDateTime, ssTime, ssCustom )
7002:   TSortDirection', '( sdAscending, sdDescending )
7003:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7004:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
7005:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7006:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7007:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7008:   SIRegister_TSortOptions(CL);
7009:   SIRegister_TPrintOptions(CL);
7010:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7011:   SIRegister_TSortedList(CL);
7012:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7013:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7014:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7015:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7016:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7017:   SIRegister_TFontSetting(CL);
7018:   SIRegister_TFontList(CL);
7019:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
7020:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7021:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7022:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7023:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
7024:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7025:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7026:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7027:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
7028:    +'r; var SortStyle : TSortStyle)
7029:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
7030:    +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7031:   SIRegister_TSortGrid(CL);
7032:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
7033:  Function NormalCompare( const Str1, Str2 : String) : Integer
7034:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
7035:  Function NumericCompare( const Str1, Str2 : String) : Integer
7036:  Function TimeCompare( const Str1, Str2 : String) : Integer
7037:  //Function Compare( Item1, Item2 : Pointer) : Integer
7038: end;
7039:
7040: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
7041:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
7042:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
7043:  Procedure IBDataBaseError
7044:  Function StatusVector : PISC_STATUS
7045:  Function StatusVectorArray : PStatusVector
7046:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
7047:  Function StatusVectorAsText : string
7048:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7049:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7050:
7051:
7052: //****************************unit uPSI_BoldUtils;*****************************************
7053:  Function CharCount( c : char; const s : string) : integer
7054:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
7055:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7056:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7057:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7058:  Function BoldTrim( const S : string) : string
7059:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
7060:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7061:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7062:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7063:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7064:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7065:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7066:  Function CapitalisedToSpaced( Capitalised : String) : String
```

```
7067:  Function SpacedToCapitalised( Spaced : String) : String
7068:  Function BooleanToString( BoolValue : Boolean) : String
7069:  Function StringToBoolean( StrValue : String) : Boolean
7070:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7071:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7072:  Function StringListToVarArray( List : TStringList) : variant
7073:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7074:  Function GetComputerNameStr : string
7075:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7076:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7077:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7078:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7079:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7080:  Procedure EnsureTrailing( var Str : String; ch : char)
7081:  Function BoldDirectoryExists( const Name : string) : Boolean
7082:  Function BoldForceDirectories( Dir : string) : Boolean
7083:  Function BoldRootRegistryKey : string
7084:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7085:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7086:  Function LogicalAnd( A, B : Integer) : Boolean
7087:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7088:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7089:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7090:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7091:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7092:  Function IsFirstInstance : Boolean
7093:  Procedure ActivateFirst( AString : PChar)
7094:  Procedure ActivateFirstCommandLine
7095:  function MakeAckPkt(const BlockNumber: Word): string;
7096:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7097:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7098:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7099:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7100:  function IdStrToWord(const Value: String): Word;
7101:  function IdWordToStr(const Value: Word): WordStr;
7102:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7103:  Function CPUFeatures : TCPUFeatures
7104:
7105: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7106: begin
7107:   AddTypeS('TXRTLBitIndex', 'Integer
7108:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7109:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7110:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7111:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7112:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7113:  Function XRTLSwapHiLo16( X : Word) : Word
7114:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7115:  Function XRTLSwapHiLo64( X : Int64) : Int64
7116:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7117:  Function XRTLROR32( A, S : Cardinal) : Cardinal
7118:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7119:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7120:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7121:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7122:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7123:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7124:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7125:  Function XRTLPopulation( A : Cardinal) : Cardinal
7126: end;
7127:
7128: Function XRTLURLDecode( const ASrc : WideString) : WideString
7129: Function XRTLURLEncode( const ASrc : WideString) : WideString
7130: Function XRTLURINormalize( const AURI : WideString) : WideString
7131: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7132: Function XRTLExtractLongPathName(APath: string): string;
7133:
7134: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7135: begin
7136:   AddTypeS('Int8', 'ShortInt
7137:   AddTypeS('Int16', 'SmallInt
7138:   AddTypeS('Int32', 'LongInt
7139:   AddTypeS('UInt8', 'Byte
7140:   AddTypeS('UInt16', 'Word
7141:   AddTypeS('UInt32', 'LongWord
7142:   AddTypeS('UInt64', 'Int64
7143:   AddTypeS('Word8', 'UInt8
7144:   AddTypeS('Word16', 'UInt16
7145:   AddTypeS('Word32', 'UInt32
7146:   AddTypeS('Word64', 'UInt64
7147:   AddTypeS('LargeInt', 'Int64
7148:   AddTypeS('NativeInt', 'Integer
7149:   AddTypeS('NativeUInt', 'Cardinal
7150:   Const('BitsPerByte','LongInt'( 8);
7151:   Const('BitsPerWord','LongInt'( 16);
7152:   Const('BitsPerLongWord','LongInt'( 32);
7153:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
```

```
7154:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7155:  Function MinI( const A, B : Integer) : Integer
7156:  Function MaxI( const A, B : Integer) : Integer
7157:  Function MinC( const A, B : Cardinal) : Cardinal
7158:  Function MaxC( const A, B : Cardinal) : Cardinal
7159:  Function SumClipI( const A, I : Integer) : Integer
7160:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7161:  Function InByteRange( const A : Int64) : Boolean
7162:  Function InWordRange( const A : Int64) : Boolean
7163:  Function InLongWordRange( const A : Int64) : Boolean
7164:  Function InShortIntRange( const A : Int64) : Boolean
7165:  Function InSmallIntRange( const A : Int64) : Boolean
7166:  Function InLongIntRange( const A : Int64) : Boolean
7167:   AddTypeS('Bool8', 'ByteBool
7168:   AddTypeS('Bool16', 'WordBool
7169:   AddTypeS('Bool32', 'LongBool
7170:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7171:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7172:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7173:  Const('MinSingle','Single').setExtended( 1.5E-45);
7174:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7175:  Const('MinDouble','Double').setExtended( 5.0E-324);
7176:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7177:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7178:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7179:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7180:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7181:  Function MinF( const A, B : Float) : Float
7182:  Function MaxF( const A, B : Float) : Float
7183:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7184:  Function InSingleRange( const A : Float) : Boolean
7185:  Function InDoubleRange( const A : Float) : Boolean
7186:  Function InCurrencyRange( const A : Float) : Boolean;
7187:  Function InCurrencyRange1( const A : Int64) : Boolean;
7188:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7189:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7190:  Function FloatIsInfinity( const A : Extended) : Boolean
7191:  Function FloatIsNaN( const A : Extended) : Boolean
7192:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7193:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7194:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7195:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7196:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7197:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7198:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7199:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7200:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7201:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7202:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7203:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7204:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7205:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7206:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7207:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7208:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7209:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7210:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7211:  Function SetBitScanForward( const Value : LongWord) : Integer;
7212:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7213:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7214:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7215:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7216:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7217:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7218:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7219:  Function cReverseBits( const Value : LongWord) : LongWord;
7220:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7221:  Function cSwapEndian( const Value : LongWord) : LongWord
7222:  Function cTwosComplement( const Value : LongWord) : LongWord
7223:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7224:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7225:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7226:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7227:  Function cBitCount( const Value : LongWord) : LongWord
7228:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7229:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7230:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7231:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7232:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7233:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7234:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7235:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7236:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7237: //  AddTypeS('CharSet', 'set of AnsiChar
7238:   AddTypeS('CharSet', 'set of Char       //!!!
7239:   AddTypeS('AnsiCharSet', 'TCharSet
7240:   AddTypeS('ByteSet', 'set of Byte
7241:   AddTypeS('AnsiChar', 'Char
7242:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
```

```
7243:  Function AsByteSet( const C : array of Byte) : ByteSet
7244:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7245:  Procedure ClearCharSet( var C : CharSet)
7246:  Procedure FillCharSet( var C : CharSet)
7247:  procedure FillCharSearchRec;  // with 0
7248:  Procedure ComplementCharSet( var C : CharSet)
7249:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7250:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7251:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7252:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7253:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7254:  Function IsSubSet( const A, B : CharSet) : Boolean
7255:  Function IsEqual( const A, B : CharSet) : Boolean
7256:  Function IsEmpty( const C : CharSet) : Boolean
7257:  Function IsComplete( const C : CharSet) : Boolean
7258:  Function cCharCount( const C : CharSet) : Integer
7259:  Procedure ConvertCaseInsensitive( var C : CharSet)
7260:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7261:  Function IntRangeLength( const Low, High : Integer) : Int64
7262:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7263:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7264:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7265:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7266:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7267:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7268:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7269:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7270:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7271:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7272:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7273:   AddTypeS('UnicodeChar', 'WideChar
7274:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7275:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7276:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7277:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7278:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7279:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7280:  Function cSgn( const A : LongInt) : Integer;
7281:  Function cSgn1( const A : Int64) : Integer;
7282:  Function cSgn2( const A : Extended) : Integer;
7283:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7284:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7285:  Function WideCharToInt( const A : WideChar) : Integer
7286:  Function CharToInt( const A : Char) : Integer
7287:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7288:  Function IntToWideChar( const A : Integer) : WideChar
7289:  Function IntToChar( const A : Integer) : Char
7290:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7291:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7292:  Function IsHexChar( const Ch : Char) : Boolean
7293:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7294:  Function HexWideCharToInt( const A : WideChar) : Integer
7295:  Function HexCharToInt( const A : Char) : Integer
7296:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7297:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7298:  Function IntToUpperHexChar( const A : Integer) : Char
7299:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7300:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
7301:  Function IntToLowerHexChar( const A : Integer) : Char
7302:  Function IntToStringA( const A : Int64) : AnsiString
7303:  Function IntToStringW( const A : Int64) : WideString
7304:  Function IntToString( const A : Int64) : String
7305:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7306:  Function UIntToStringW( const A : NativeUInt) : WideString
7307:  Function UIntToString( const A : NativeUInt) : String
7308:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7309:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7310:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7311:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7312:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7313:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7314:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7315:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7316:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7317:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7318:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7319:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7320:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7321:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7322:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7323:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7324:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7325:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7326:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7327:  Function StringToInt64A( const S : AnsiString) : Int64
7328:  Function StringToInt64W( const S : WideString) : Int64
7329:  Function StringToInt64( const S : String) : Int64
7330:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7331:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
```

```
7332:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7333:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7334:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7335:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7336:  Function StringToIntA( const S : AnsiString) : Integer
7337:  Function StringToIntW( const S : WideString) : Integer
7338:  Function StringToInt( const S : String) : Integer
7339:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7340:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7341:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7342:  Function StringToLongWordA( const S : AnsiString) : LongWord
7343:  Function StringToLongWordW( const S : WideString) : LongWord
7344:  Function StringToLongWord( const S : String) : LongWord
7345:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7346:  Function HexToUIntW( const S : WideString) : NativeUInt
7347:  Function HexToUInt( const S : String) : NativeUInt
7348:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7349:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7350:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7351:  Function HexToLongWordA( const S : AnsiString) : LongWord
7352:  Function HexToLongWordW( const S : WideString) : LongWord
7353:  Function HexToLongWord( const S : String) : LongWord
7354:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7355:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7356:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7357:  Function OctToLongWordA( const S : AnsiString) : LongWord
7358:  Function OctToLongWordW( const S : WideString) : LongWord
7359:  Function OctToLongWord( const S : String) : LongWord
7360:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7361:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7362:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7363:  Function BinToLongWordA( const S : AnsiString) : LongWord
7364:  Function BinToLongWordW( const S : WideString) : LongWord
7365:  Function BinToLongWord( const S : String) : LongWord
7366:  Function FloatToStringA( const A : Extended) : AnsiString
7367:  Function FloatToStringW( const A : Extended) : WideString
7368:  Function FloatToString( const A : Extended) : String
7369:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7370:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7371:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7372:  Function StringToFloatA( const A : AnsiString) : Extended
7373:  Function StringToFloatW( const A : WideString) : Extended
7374:  Function StringToFloat( const A : String) : Extended
7375:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7376:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7377:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7378:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
      PadChar: AnsiChar) : AnsiString
7379:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7380:  unit uPSI_cFundamentUtils;
7381:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7382:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7383:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7384:  Const('CCHARSET','Stringb64_XXEncode);
7385:  Const('CHEXSET','String'0123456789ABCDEF
7386:  Const('HEXDIGITS','String'0123456789ABCDEF
7387:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7388:  Const('DIGISET','String'0123456789
7389:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7390:  Const('DIGISET2','TCharset').SetSet('0123456789'
7391:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7392:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7393:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7394:  Const('NUMBERS','String'0123456789');
7395:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7396:  Function CharSetToStr( const C : CharSet) : AnsiString
7397:  Function StrToCharSet( const S : AnsiString) : CharSet
7398:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7399:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7400:  Function UUDecode( const S : AnsiString) : AnsiString
7401:  Function XXDecode( const S : AnsiString) : AnsiString
7402:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7403:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7404:  Function InterfaceToStrW( const I : IInterface) : WideString
7405:  Function InterfaceToStr( const I : IInterface) : String
7406:  Function ObjectClassName( const O : TObject) : String
7407:  Function ClassClassName( const C : TClass) : String
7408:  Function ObjectToStr( const O : TObject) : String
7409:  Function ObjectToString( const O : TObject) : String
7410:  Function CharSetToStr( const C : CharSet) : AnsiString
7411:  Function StrToCharSet( const S : AnsiString) : CharSet
7412:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7413:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7414:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7415:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7416:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
```

```
7417:  Const('Bytes1KB','LongInt'( 1024);
7418:   SIRegister_IInterface(CL);
7419:  Procedure SelfTestCFundamentUtils
7420:
7421: Function CreateSchedule : IJclSchedule
7422: Function NullStamp : TTimeStamp
7423: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7424: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7425: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7426:
7427: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7428: begin
7429:  AddTypeS('TFunc', 'function(X : Float) : Float;
7430: Function InitGraphics( Width, Height : Integer) : Boolean
7431: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7432: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7433: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7434: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7435: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7436: Procedure SetGraphTitle( Title : String)
7437: Procedure SetOxTitle( Title : String)
7438: Procedure SetOyTitle( Title : String)
7439: Function GetGraphTitle : String
7440: Function GetOxTitle : String
7441: Function GetOyTitle : String
7442: Procedure PlotOxAxis( Canvas : TCanvas)
7443: Procedure PlotOyAxis( Canvas : TCanvas)
7444: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7445: Procedure WriteGraphTitle( Canvas : TCanvas)
7446: Function SetMaxCurv( NCurv : Byte) : Boolean
7447: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7448: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7449: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7450: Procedure SetCurvStep( CurvIndex, Step : Integer)
7451: Function GetMaxCurv : Byte
7452: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7453: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7454: Function GetCurvLegend( CurvIndex : Integer) : String
7455: Function GetCurvStep( CurvIndex : Integer) : Integer
7456: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7457: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7458: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7459: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7460: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7461: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7462: Function Xpixel( X : Float) : Integer
7463: Function Ypixel( Y : Float) : Integer
7464: Function Xuser( X : Integer) : Float
7465: Function Yuser( Y : Integer) : Float
7466: end;
7467:
7468: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7469: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7470: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7471: Procedure FFT_Integer_Cleanup
7472: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7473: //unit uPSI_JclStreams;
7474: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7475: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7476: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7477: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7478:
7479: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7480: begin
7481:  FindClass('TOBJECT'),'EInvalidDest
7482:  FindClass('TOBJECT'),'EFCantMove
7483:  Procedure fmxCopyFile( const FileName, DestName : string)
7484:  Procedure fmxMoveFile( const FileName, DestName : string)
7485:  Function fmxGetFileSize( const FileName : string) : LongInt
7486:  Function fmxFileDateTime( const FileName : string) : TDateTime
7487:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7488:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7489: end;
7490:
7491: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7492: begin
7493:  SIRegister_IFindFileIterator(CL);
7494:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7495: end;
7496:
7497: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7498: begin
7499:  Function SkipWhite( cp : PChar) : PChar
7500:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7501:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7502:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7503: end;
7504:
7505: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
```

```
7506: begin
7507:   SIRegister_TStringHashMapTraits(CL);
7508:  Function CaseSensitiveTraits : TStringHashMapTraits
7509:  Function CaseInsensitiveTraits : TStringHashMapTraits
7510:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7511:    +'e; Right : PHashNode; end
7512:   //PHashArray', '^THashArray // will not work
7513:   SIRegister_TStringHashMap(CL);
7514:  THashValue', 'Cardinal
7515:  Function StrHash( const s : string) : THashValue
7516:  Function TextHash( const s : string) : THashValue
7517:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7518:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7519:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7520:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7521:   SIRegister_TCaseSensitiveTraits(CL);
7522:   SIRegister_TCaseInsensitiveTraits(CL);
7523:
7524:
7525: //************************************************************************unit uPSI_umath;
7526:  Function uExpo( X : Float) : Float
7527:  Function uExp2( X : Float) : Float
7528:  Function uExp10( X : Float) : Float
7529:  Function uLog( X : Float) : Float
7530:  Function uLog2( X : Float) : Float
7531:  Function uLog10( X : Float) : Float
7532:  Function uLogA( X, A : Float) : Float
7533:  Function uIntPower( X : Float; N : Integer): Float
7534:  Function uPower( X, Y : Float) : Float
7535:  Function SgnGamma( X : Float) : Integer
7536:  Function Stirling( X : Float) : Float
7537:  Function StirLog( X : Float) : Float
7538:  Function Gamma( X : Float) : Float
7539:  Function LnGamma( X : Float) : Float
7540:  Function DiGamma( X : Float) : Float
7541:  Function TriGamma( X : Float) : Float
7542:  Function IGamma( X : Float) : Float
7543:  Function JGamma( X : Float) : Float
7544:  Function InvGamma( X : Float) : Float
7545:  Function Erf( X : Float) : Float
7546:  Function Erfc( X : Float) : Float
7547: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7548: { Correlation coefficient between samples X and Y }
7549: function DBeta(A, B, X : Float) : Float;
7550: { Density of Beta distribution with parameters A and B }
7551: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7552: Function Beta(X, Y : Float) : Float
7553: Function Binomial( N, K : Integer) : Float
7554: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7555: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7556: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7557: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7558: Function DNorm( X : Float) : Float
7559:
7560: function DGamma(A, B, X : Float) : Float;
7561: { Density of Gamma distribution with parameters A and B }
7562: function DKhi2(Nu : Integer; X : Float) : Float;
7563: { Density of Khi-2 distribution with Nu d.o.f. }
7564: function DStudent(Nu : Integer; X : Float) : Float;
7565: { Density of Student distribution with Nu d.o.f. }
7566: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7567: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7568: function IBeta(A, B, X : Float) : Float;
7569: { Incomplete Beta function}
7570: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7571:
7572: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7573: begin
7574:  Procedure SetOptAlgo( Algo : TOptAlgo)
7575: procedure SetOptAlgo(Algo : TOptAlgo);
7576: { ----------------------------------------------------------------
7577:   Sets the optimization algorithm according to Algo, which must be
7578:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7579:
7580:  Function GetOptAlgo : TOptAlgo
7581:  Procedure SetMaxParam( N : Byte)
7582:  Function GetMaxParam : Byte
7583:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7584:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7585:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7586:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
      Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7587:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
      MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7588:  Procedure SetMCFile( FileName : String)
7589:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7590:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
      LastPar:Integer;V:TMatrix);
7591: end;
```

```
7592:
7593: (*----------------------------------------------------------------------------*)
7594: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7595: begin
7596:  Procedure SaveSimplex( FileName : string)
7597:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7598: end;
7599: (*----------------------------------------------------------------------------*)
7600: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7601: begin
7602:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7603:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7604: end;
7605:
7606: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7607: begin
7608:  Function LTrim( S : String) : String
7609:  Function RTrim( S : String) : String
7610:  Function uTrim( S : String) : String
7611:  Function StrChar( N : Byte; C : Char) : String
7612:  Function RFill( S : String; L : Byte) : String
7613:  Function LFill( S : String; L : Byte) : String
7614:  Function CFill( S : String; L : Byte) : String
7615:  Function Replace( S : String; C1, C2 : Char) : String
7616:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7617:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7618:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7619:  Function FloatStr( X : Float) : String
7620:  Function IntStr( N : LongInt) : String
7621:  Function uCompStr( Z : Complex) : String
7622: end;
7623:
7624: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7625: begin
7626:  Function uSinh( X : Float) : Float
7627:  Function uCosh( X : Float) : Float
7628:  Function uTanh( X : Float) : Float
7629:  Function uArcSinh( X : Float) : Float
7630:  Function uArcCosh( X : Float) : Float
7631:  Function ArcTanh( X : Float) : Float
7632:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7633: end;
7634:
7635: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7636: begin
7637: type RNG_Type =
7638:   (RNG_MWC,      { Multiply-With-Carry }
7639:    RNG_MT,       { Mersenne Twister }
7640:    RNG_UVAG);    { Universal Virtual Array Generator }
7641:  Procedure SetRNG( RNG : RNG_Type)
7642:  Procedure InitGen( Seed : RNG_IntType)
7643:  Procedure SRand( Seed : RNG_IntType)
7644:  Function IRanGen : RNG_IntType
7645:  Function IRanGen31 : RNG_IntType
7646:  Function RanGen1 : Float
7647:  Function RanGen2 : Float
7648:  Function RanGen3 : Float
7649:  Function RanGen53 : Float
7650: end;
7651:
7652: //  Optimization by Simulated Annealing
7653: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7654: begin
7655:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7656:  Procedure SA_CreateLogFile( FileName : String)
7657:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7658: end;
7659:
7660: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7661: begin
7662:  Procedure InitUVAGbyString( KeyPhrase : string)
7663:  Procedure InitUVAG( Seed : RNG_IntType)
7664:  Function IRanUVAG : RNG_IntType
7665: end;
7666:
7667: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7668: begin
7669:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7670:  Procedure GA_CreateLogFile( LogFileName : String)
7671:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7672: end;
7673:
7674:  TVector', 'array of Float
7675: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7676: begin
7677:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7678:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7679: end;
7680:
```

```
7681: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7682: begin
7683:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7684:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7685: end;
7686:
7687: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7688: begin
7689:   FT_Result', 'Integer
7690:   //TDWordptr', '^DWord // will not work
7691:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7692:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7693:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7694:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7695:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7696:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7697:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7698:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7699:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7700:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7701:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7702:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7703:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7704:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7705:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7706:    yte; end
7707: end;
7708:
7709:
7710: //************************************ PaintFX*************************
7711: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7712: begin
7713:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7714:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7715:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7716:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7717:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7718:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7719:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7720:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7721:     Procedure Turn( Src, Dst : TBitmap)
7722:     Procedure TurnRight( Src, Dst : TBitmap)
7723:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7724:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7725:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7726:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7727:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7728:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7729:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7730:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7731:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7732:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7733:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7734:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7735:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7736:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7737:     Procedure Emboss( var Bmp : TBitmap)
7738:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7739:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7740:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7741:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7742:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7743:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7744:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7745:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7746:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7747:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7748:     Procedure SemiOpaque( Src, Dst : TBitmap)
7749:     Procedure ShadowDownLeft( const Dst : TBitmap)
7750:     Procedure ShadowDownRight( const Dst : TBitmap)
7751:     Procedure ShadowUpLeft( const Dst : TBitmap)
7752:     Procedure ShadowUpRight( const Dst : TBitmap)
7753:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7754:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7755:     Procedure FlipRight( const Dst : TBitmap)
7756:     Procedure FlipDown( const Dst : TBitmap)
7757:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7758:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7759:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7760:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7761:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7762:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7763:     Procedure SmoothResize( var Src, Dst : TBitmap)
7764:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7765:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7766:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7767:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7768:     Procedure GrayScale( const Dst : TBitmap)
7769:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
```

```
7770:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7771:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7772:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7773:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7774:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7775:     Procedure AntiAlias( const Dst : TBitmap)
7776:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7777:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7778:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7779:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7780:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7781:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7782:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7783:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7784:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7785:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7786:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7787:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7788:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7789:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7790:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7791:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7792:     Procedure Invert( Src : TBitmap)
7793:     Procedure MirrorRight( Src : TBitmap)
7794:     Procedure MirrorDown( Src : TBitmap)
7795:   end;
7796: end;
7797:
7798: (*----------------------------------------------------------------------------*)
7799: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7800: begin
7801:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7802:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7803:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7804:   SIRegister_TJvPaintFX(CL);
7805:  Function SplineFilter( Value : Single) : Single
7806:  Function BellFilter( Value : Single) : Single
7807:  Function TriangleFilter( Value : Single) : Single
7808:  Function BoxFilter( Value : Single) : Single
7809:  Function HermiteFilter( Value : Single) : Single
7810:  Function Lanczos3Filter( Value : Single) : Single
7811:  Function MitchellFilter( Value : Single) : Single
7812: end;
7813:
7814:
7815: (*----------------------------------------------------------------------------*)
7816: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7817: begin
7818:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7819:  TeeMsg_DefaultSeriesName','String 'Series
7820:  TeeMsg_DefaultToolName','String 'ChartTool
7821:  ChartComponentPalette','String 'TeeChart
7822:  TeeMaxLegendColumns',LongInt'( 2);
7823:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7824:  TeeTitleFootDistance,LongInt( 5);
7825:   SIRegister_TCustomChartWall(CL);
7826:   SIRegister_TChartWall(CL);
7827:   SIRegister_TChartLegendGradient(CL);
7828:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7829:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7830:   FindClass('TOBJECT'),'LegendException
7831:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7832:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7833:   FindClass('TOBJECT'),'TCustomChartLegend
7834:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7835:   TLegendSymbolPosition', '( spLeft, spRight )
7836:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7837:   TSymbolCalcHeight', 'Function  : Integer
7838:   SIRegister_TLegendSymbol(CL);
7839:   SIRegister_TTeeCustomShapePosition(CL);
7840:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7841:   SIRegister_TLegendTitle(CL);
7842:   SIRegister_TLegendItem(CL);
7843:   SIRegister_TLegendItems(CL);
7844:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7845:   FindClass('TOBJECT'),'TCustomChart
7846:   SIRegister_TCustomChartLegend(CL);
7847:   SIRegister_TChartLegend(CL);
7848:   SIRegister_TChartTitle(CL);
7849:   SIRegister_TChartFootTitle(CL);
7850:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7851:    +'eButton; Shift : TShiftState; X, Y : Integer)
7852:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7853:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7854:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7855:    +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7856:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7857:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7858:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
```

```
7859:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7860:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7861:    +'toMax : Boolean; Min : Double; Max : Double; end
7862:   TAllAxisSavedScales', 'array of TAxisSavedScales
7863:   SIRegister_TChartBackWall(CL);
7864:   SIRegister_TChartRightWall(CL);
7865:   SIRegister_TChartBottomWall(CL);
7866:   SIRegister_TChartLeftWall(CL);
7867:   SIRegister_TChartWalls(CL);
7868:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7869:   SIRegister_TCustomChart(CL);
7870:   SIRegister_TChart(CL);
7871:   SIRegister_TTeeSeriesTypes(CL);
7872:   SIRegister_TTeeToolTypes(CL);
7873:   SIRegister_TTeeDragObject(CL);
7874:   SIRegister_TColorPalettes(CL);
7875:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Integer;
7876:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7877:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries: Int;
7878:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7879:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7880:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7881:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7882:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7883:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7884:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
7885:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7886:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7887:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7888:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7889:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7890:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7891:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7892:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7893:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7894:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7895:  Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
      R:TRect;ReferenceChart:TCustomChart);
7896:   SIRegister_TChartTheme(CL);
7897:   //TChartThemeClass', 'class of TChartTheme
7898:   //TCanvasClass', 'class of TCanvas3D
7899:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7900:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7901:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7902:  Procedure ShowMessageUser( const S : String)
7903:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7904:  Function HasLabels( ASeries : TChartSeries) : Boolean
7905:  Function HasColors( ASeries : TChartSeries) : Boolean
7906:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7907:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7908: end;
7909:
7910:
7911: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7912: begin
7913:  //'TeeFormBorderStyle',' bsNone);
7914:   SIRegister_TMetafile(CL);
7915:  'TeeDefVerticalMargin','LongInt'( 4);
7916:  'TeeDefHorizMargin','LongInt'( 3);
7917:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7918:  'TeeMsg_TeeHand','String 'crTeeHand
7919:  'TeeNormalPrintDetail','LongInt'( 0);
7920:  'TeeHighPrintDetail','LongInt'( - 100);
7921:  'TeeDefault_PrintMargin','LongInt'( 15);
7922:  'MaxDefaultColors','LongInt'( 19);
7923:  'TeeTabDelimiter','Char #9);
7924:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7925:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7926:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7927:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7928:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7929:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7930:   SIRegister_TCustomPanelNoCaption(CL);
7931:   FindClass('TOBJECT'),'TCustomTeePanel
7932:   SIRegister_TZoomPanning(CL);
7933:   SIRegister_TTeeEvent(CL);
7934:   //SIRegister_TTeeEventListeners(CL);
7935:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7936:   SIRegister_TTeeMouseEvent(CL);
7937:   SIRegister_TCustomTeePanel(CL);
7938:   //TChartGradient', 'TTeeGradient
7939:   //TChartGradientClass', 'class of TChartGradient
7940:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7941:   SIRegister_TTeeZoomPen(CL);
7942:   SIRegister_TTeeZoomBrush(CL);
```

```
7943:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7944:   SIRegister_TTeeZoom(CL);
7945:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7946:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7947:   SIRegister_TBackImage(CL);
7948:   SIRegister_TCustomTeePanelExtended(CL);
7949:   //TChartBrushClass', 'class of TChartBrush
7950:   SIRegister_TTeeCustomShapeBrushPen(CL);
7951:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7952:   TTextFormat', '( ttfNormal, ttfHtml )
7953:   SIRegister_TTeeCustomShape(CL);
7954:   SIRegister_TTeeShape(CL);
7955:   SIRegister_TTeeExportData(CL);
7956:  Function TeeStr( const Num : Integer) : String
7957:  Function DateTimeDefaultFormat( const AStep : Double) : String
7958:  Function TEEDaysInMonth( Year, Month : Word) : Word
7959:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7960:  Function NextDateTimeStep( const AStep : Double) : Double
7961:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7962:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7963:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7964:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7965:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7966:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7967:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7968:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7969:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7970:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7971:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7972:  Function DelphiToLocalFormat( const Format : String) : String
7973:  Function LocalToDelphiFormat( const Format : String) : String
7974:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
7975:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7976:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7977:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
7978:   TTeeSortSwap', 'Procedure ( a, b : Integer)
7979:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
7980:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
7981:  Function TeeExtractField( St : String; Index : Integer) : String;
7982:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7983:  Function TeeNumFields( St : String) : Integer;
7984:  Function TeeNumFields1( const St, Separator : String) : Integer;
7985:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7986:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7987:  // TColorArray', 'array of TColor
7988:  Function GetDefaultColor( const Index : Integer) : TColor
7989:  Procedure SetDefaultColorPalette;
7990:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
7991:  'TeeCheckBoxSize','LongInt'( 11);
7992:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
7993:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
7994:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
7995:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
7996:  Procedure TeeTranslateControl( AControl : TControl);
7997:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
7998:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
7999:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8000:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8001:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8002:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8003:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8004:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8005:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8006:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8007:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8008:  Procedure TeeSaveStringOption( const AKey, Value : String)
8009:  Function TeeDefaultXMLEncoding : String
8010:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8011:   TeeWindowHandle', 'Integer
8012:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
8013:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8014:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8015: end;
8016:
8017:
8018: using mXBDEUtils
8019: ********************************************************************************
8020:  Procedure SetAlias( aAlias, aDirectory : String)
8021:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
8022:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
8023:  Procedure SetBDE( aPath, aNode, aValue : String)
8024:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8025:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8026:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8027:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8028:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8029:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
```

```
8030:
8031:
8032: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
8033: begin
8034: AddClassN(FindClass('TOBJECT'),'EDateTime
8035: Function DatePart( const D : TDateTime) : Integer
8036: Function TimePart( const D : TDateTime) : Double
8037: Function Century( const D : TDateTime) : Word
8038: Function Year( const D : TDateTime) : Word
8039: Function Month( const D : TDateTime) : Word
8040: Function Day( const D : TDateTime) : Word
8041: Function Hour( const D : TDateTime) : Word
8042: Function Minute( const D : TDateTime) : Word
8043: Function Second( const D : TDateTime) : Word
8044: Function Millisecond( const D : TDateTime) : Word
8045: ('OneDay','Extended').setExtended( 1.0);
8046: ('OneHour','Extended').SetExtended( OneDay / 24);
8047: ('OneMinute','Extended').SetExtended( OneHour / 60);
8048: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8049: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8050: ('OneWeek','Extended').SetExtended( OneDay * 7);
8051: ('HoursPerDay','Extended').SetExtended( 24);
8052: ('MinutesPerHour','Extended').SetExtended( 60);
8053: ('SecondsPerMinute','Extended').SetExtended( 60);
8054: Procedure SetYear( var D : TDateTime; const Year : Word)
8055: Procedure SetMonth( var D : TDateTime; const Month : Word)
8056: Procedure SetDay( var D : TDateTime; const Day : Word)
8057: Procedure SetHour( var D : TDateTime; const Hour : Word)
8058: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8059: Procedure SetSecond( var D : TDateTime; const Second : Word)
8060: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8061: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8062: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8063: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8064: Function IsAM( const D : TDateTime) : Boolean
8065: Function IsPM( const D : TDateTime) : Boolean
8066: Function IsMidnight( const D : TDateTime) : Boolean
8067: Function IsNoon( const D : TDateTime) : Boolean
8068: Function IsSunday( const D : TDateTime) : Boolean
8069: Function IsMonday( const D : TDateTime) : Boolean
8070: Function IsTuesday( const D : TDateTime) : Boolean
8071: Function IsWedneday( const D : TDateTime) : Boolean
8072: Function IsThursday( const D : TDateTime) : Boolean
8073: Function IsFriday( const D : TDateTime) : Boolean
8074: Function IsSaturday( const D : TDateTime) : Boolean
8075: Function IsWeekend( const D : TDateTime) : Boolean
8076: Function Noon( const D : TDateTime) : TDateTime
8077: Function Midnight( const D : TDateTime) : TDateTime
8078: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8079: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8080: Function NextWorkday( const D : TDateTime) : TDateTime
8081: Function PreviousWorkday( const D : TDateTime) : TDateTime
8082: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8083: Function LastDayOfYear( const D : TDateTime) : TDateTime
8084: Function EasterSunday( const Year : Word) : TDateTime
8085: Function GoodFriday( const Year : Word) : TDateTime
8086: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8087: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8088: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8089: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8090: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8091: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8092: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8093: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8094: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8095: Function DayOfYear( const D : TDateTime) : Integer
8096: Function DaysInMonth( const Ye, Mo : Word) : Integer
8097: Function DaysInMonth( const D : TDateTime) : Integer
8098: Function DaysInYear( const Ye : Word) : Integer
8099: Function DaysInYearDate( const D : TDateTime) : Integer
8100: Function WeekNumber( const D : TDateTime) : Integer
8101: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8102: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8103: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8104: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8105: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8106: Function DiffHours( const D1, D2 : TDateTime) : Integer
8107: Function DiffDays( const D1, D2 : TDateTime) : Integer
8108: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8109: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8110: Function DiffYears( const D1, D2 : TDateTime) : Integer
8111: Function GMTBias : Integer
8112: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8113: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8114: Function NowAsGMTTime : TDateTime
8115: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8116: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8117: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8118: Function DateTimeToANSI( const D : TDateTime) : Integer
```

```
8119: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8120: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8121: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8122: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8123: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8124: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8125: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8126: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8127: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8128: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8129: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8130: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8131: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8132: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8133: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8134: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8135: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8136: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8137: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8138: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8139: Function EnglishShortMonthA( const S : AnsiString) : Integer
8140: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8141: Function EnglishLongMonthA( const S : AnsiString) : Integer
8142: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8143: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8144: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8145: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8146: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8147: Function RFCMonthA( const S : AnsiString) : Word
8148: Function RFCMonthU( const S : UnicodeString) : Word
8149: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
8150: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8151: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8152: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8153: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8154: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8155: Function NowAsRFCDateTimeA : AnsiString
8156: Function NowAsRFCDateTimeU : UnicodeString
8157: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8158: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8159: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8160: Function TimePeriodStr( const D : TDateTime) : AnsiString
8161: Procedure SelfTest
8162: end;
8163: //****************************************CFileUtils
8164: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8165: Function PathHasDriveLetter( const Path : String) : Boolean
8166: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8167: Function PathIsDriveLetter( const Path : String) : Boolean
8168: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8169: Function PathIsDriveRoot( const Path : String) : Boolean
8170: Function PathIsRootA( const Path : AnsiString) : Boolean
8171: Function PathIsRoot( const Path : String) : Boolean
8172: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8173: Function PathIsUNCPath( const Path : String) : Boolean
8174: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8175: Function PathIsAbsolute( const Path : String) : Boolean
8176: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8177: Function PathIsDirectory( const Path : String) : Boolean
8178: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8179: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8180: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8181: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8182: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8183: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8184: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8185: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8186: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8187: Function PathCanonical( const Path : String; const PathSep : Char) : String
8188: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8189: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8190: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8191: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8192: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8193: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8194: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8195: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8196: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8197: Function FileNameValid( const FileName : String) : String
8198: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8199: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8200: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8201: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8202: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8203: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8204: Procedure CCopyFile( const FileName, DestName : String)
8205: Procedure CMoveFile( const FileName, DestName : String)
8206: Function CDeleteFiles( const FileMask : String) : Boolean
8207: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
```

```
8208: Procedure FileCloseEx( const FileHandle : TFileHandle)
8209: Function FileExistsA( const FileName : AnsiString) : Boolean
8210:  Function CFileExists( const FileName : String) : Boolean
8211:  Function CFileGetSize( const FileName : String) : Int64
8212: Function FileGetDateTime( const FileName : String) : TDateTime
8213: Function FileGetDateTime2( const FileName : String) : TDateTime
8214: Function FileIsReadOnly( const FileName : String) : Boolean
8215: Procedure FileDeleteEx( const FileName : String)
8216: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8217: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
      : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8218: Function DirectoryEntryExists( const Name : String) : Boolean
8219: Function DirectoryEntrySize( const Name : String) : Int64
8220:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8221: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8222: Procedure CDirectoryCreate( const DirectoryName : String)
8223: Function GetFirstFileNameMatching( const FileMask : String) : String
8224: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8225: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8226: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8227:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8228:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8229: Function DriveIsValid( const Drive : Char) : Boolean
8230: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8231: Function DriveFreeSpace( const Path : AnsiString) : Int64
8232:
8233: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8234: begin
8235:  AddClassN(FindClass('TOBJECT'),'ETimers
8236:  Const('TickFrequency','LongInt'( 1000);Function GetTick : LongWord
8237: Function TickDelta( const D1, D2 : LongWord) : Integer
8238: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8239:   AddTypeS('THPTimer', 'Int64
8240: Procedure StartTimer( var Timer : THPTimer)
8241: Procedure StopTimer( var Timer : THPTimer)
8242: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8243: Procedure InitStoppedTimer( var Timer : THPTimer)
8244: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8245: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8246: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8247: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8248: Function GetHighPrecisionFrequency : Int64
8249: Function GetHighPrecisionTimerOverhead : Int64
8250: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8251: Procedure SelfTestCTimer
8252: end;
8253:
8254: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8255: begin
8256:  Function RandomSeed : LongWord
8257:  Procedure AddEntropy( const Value : LongWord)
8258:  Function RandomUniform : LongWord;
8259:  Function RandomUniform1( const N : Integer) : Integer;
8260:  Function RandomBoolean : Boolean
8261:  Function RandomByte : Byte
8262:  Function RandomByteNonZero : Byte
8263:  Function RandomWord : Word
8264:  Function RandomInt64 : Int64;
8265:  Function RandomInt641( const N : Int64) : Int64;
8266:  Function RandomHex( const Digits : Integer) : String
8267:  Function RandomFloat : Extended
8268:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8269:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8270:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8271:  Function mwcRandomLongWord : LongWord
8272:  Function urnRandomLongWord : LongWord
8273:  Function moaRandomFloat : Extended
8274:  Function mwcRandomFloat : Extended
8275:  Function RandomNormalF : Extended
8276:  Procedure SelfTestCRandom
8277: end;
8278:
8279: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8280: begin
8281:  // PIntArray', '^TIntArray // will not work
8282:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8283:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8284:  Function synMax( x, y : integer) : integer
8285:  Function synMin( x, y : integer) : integer
8286:  Function synMinMax( x, mi, ma : integer) : integer
8287:  Procedure synSwapInt( var l, r : integer)
8288:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8289:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8290:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8291:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8292:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8293:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8294:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8295:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
```

```
8296:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8297:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8298:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8299:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8300:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8301:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8302:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8303:  ('C3_NONSPACING','LongInt'( 1);
8304:  'C3_DIACRITIC','LongInt'( 2);
8305:  'C3_VOWELMARK','LongInt'( 4);
8306:  ('C3_SYMBOL','LongInt'( 8);
8307:  ('C3_KATAKANA','LongWord( $0010);
8308:  ('C3_HIRAGANA','LongWord( $0020);
8309:  ('C3_HALFWIDTH','LongWord( $0040);
8310:  ('C3_FULLWIDTH','LongWord( $0080);
8311:  ('C3_IDEOGRAPH','LongWord( $0100);
8312:  ('C3_KASHIDA','LongWord( $0200);
8313:  ('C3_LEXICAL','LongWord( $0400);
8314:  ('C3_ALPHA','LongWord( $8000);
8315:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8316:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8317:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8318:  Function synIsStringType( Value : Word) : TStringType
8319:  Function synGetEOL( Line : PChar) : PChar
8320:  Function synEncodeString( s : string) : string
8321:  Function synDecodeString( s : string) : string
8322:  Procedure synFreeAndNil( var Obj: TObject)
8323:  Procedure synAssert( Expr : Boolean)
8324:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8325:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8326:   TReplaceFlags', 'set of TReplaceFlag )
8327:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8328:  Function synGetRValue( RGBValue : TColor) : byte
8329:  Function synGetGValue( RGBValue : TColor) : byte
8330:  Function synGetBValue( RGBValue : TColor) : byte
8331:  Function synRGB( r, g, b : Byte) : Cardinal
8332: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8333:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8334:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8335:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8336:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8337: end;
8338:
8339:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8340:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8341:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8342:
8343: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8344: begin
8345:  Function STimeZoneBias : integer
8346:  Function TimeZone : string
8347:  Function Rfc822DateTime( t : TDateTime) : string
8348:  Function CDateTime( t : TDateTime) : string
8349:  Function SimpleDateTime( t : TDateTime) : string
8350:  Function AnsiCDateTime( t : TDateTime) : string
8351:  Function GetMonthNumber( Value : String) : integer
8352:  Function GetTimeFromStr( Value : string) : TDateTime
8353:  Function GetDateMDYFromStr( Value : string) : TDateTime
8354:  Function DecodeRfcDateTime( Value : string) : TDateTime
8355:  Function GetUTTime : TDateTime
8356:  Function SetUTTime( Newdt : TDateTime) : Boolean
8357:  Function SGetTick : LongWord
8358:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8359:  Function CodeInt( Value : Word) : Ansistring
8360:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8361:  Function CodeLongInt( Value : LongInt) : Ansistring
8362:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8363:  Function DumpStr( const Buffer : Ansistring) : string
8364:  Function DumpExStr( const Buffer : Ansistring) : string
8365:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8366:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8367:  Function TrimSPLeft( const S : string) : string
8368:  Function TrimSPRight( const S : string) : string
8369:  Function TrimSP( const S : string) : string
8370:  Function SeparateLeft( const Value, Delimiter : string) : string
8371:  Function SeparateRight( const Value, Delimiter : string) : string
8372:  Function SGetParameter( const Value, Parameter : string) : string
8373:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8374:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8375:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8376:  Function GetEmailAddr( const Value : string) : string
8377:  Function GetEmailDesc( Value : string) : string
8378:  Function CStrToHex( const Value : Ansistring) : string
8379:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8380:  Function CBinToInt( const Value : string) : Integer
8381:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8382:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
```

```
8383:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8384:  Function CRPos( const Sub, Value : String) : Integer
8385:  Function FetchBin( var Value : string; const Delimiter : string) : string
8386:  Function CFetch( var Value : string; const Delimiter : string) : string
8387:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8388:  Function IsBinaryString( const Value : AnsiString) : Boolean
8389:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8390:  Procedure StringsTrim( const value : TStrings)
8391:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8392:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8393:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8394:  Function CCountOfChar( const Value : string; aChr : char) : integer
8395:  Function UnquoteStr( const Value : string; Quote : Char) : string
8396:  Function QuoteStr( const Value : string; Quote : Char) : string
8397:  Procedure HeadersToList( const Value : TStrings)
8398:  Procedure ListToHeaders( const Value : TStrings)
8399:  Function SwapBytes( Value : integer) : integer
8400:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8401:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8402:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8403:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8404:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8405:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8406: end;
8407:
8408: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8409: begin
8410:  ('CrcBufSize','LongInt'( 2048);
8411:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8412:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8413:  Function Adler32OfFile( FileName : AnsiString) : LongInt
8414:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8415:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8416:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8417:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8418:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8419:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8420:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8421:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8422:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8423:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8424:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8425:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8426: end;
8427:
8428: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8429: begin
8430:  function CreateOleObject(const ClassName: String): IDispatch;
8431:  function GetActiveOleObject(const ClassName: String): IDispatch;
8432:  function ProgIDToClassID(const ProgID: string): TGUID;
8433:  function ClassIDToProgID(const ClassID: TGUID): string;
8434:  function CreateClassID: string;
8435:  function CreateGUIDString: string;
8436:  function CreateGUIDID: string;
8437:  procedure OleError(ErrorCode: longint)
8438:  procedure OleCheck(Result: HResult);
8439: end;
8440:
8441:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8442:  Function xGetActiveOleObject( const ClassName : string) : Variant
8443:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8444:  Function DllCanUnloadNow : HResult
8445:  Function DllRegisterServer : HResult
8446:  Function DllUnregisterServer : HResult
8447:  Function VarFromInterface( Unknown : IUnknown) : Variant
8448:  Function VarToInterface( const V : Variant) : IDispatch
8449:  Function VarToAutoObject( const V : Variant) : TAutoObject
8450:  //Procedure
      DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8451:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8452:  Procedure OleError( ErrorCode : HResult)
8453:  Procedure OleCheck( Result : HResult)
8454:  Function StringToClassID( const S : string) : TCLSID
8455:  Function ClassIDToString( const ClassID : TCLSID) : string
8456:  Function xProgIDToClassID( const ProgID : string) : TCLSID
8457:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8458:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8459:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8460:  Function xGUIDToString( const ClassID : TGUID) : string
8461:  Function xStringToGUID( const S : string) : TGUID
8462:  Function xGetModuleName( Module : HMODULE) : string
8463:  Function xAcquireExceptionObject : TObject
8464:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8465:  Function xUtf8Encode( const WS : WideString) : UTF8String
8466:  Function xUtf8Decode( const S : UTF8String) : WideString
8467:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8468:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8469:  Function XRTLHandleCOMException : HResult
8470:  Procedure XRTLCheckArgument( Flag : Boolean)
```

```
8471:  //Procedure XRTLCheckOutArgument( out Arg)
8472:  Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8473:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8474:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8475:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8476:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8477:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8478: function   XRTLDefaultCategoryManager: IUnknown;
8479: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8480: // ICatRegister helper functions
8481: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8482:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8483:                                        const CategoryManager: IUnknown = nil): HResult;
8484: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8485:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8486:                                        const CategoryManager: IUnknown = nil): HResult;
8487: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8488:                                        const CategoryManager: IUnknown = nil): HResult;
8489: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8490:                                          const CategoryManager: IUnknown = nil): HResult;
8491: // ICatInformation helper functions
8492: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8493:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
8494:                                       const CategoryManager: IUnknown = nil): HResult;
8495: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8496:                                const CategoryManager: IUnknown = nil): HResult;
8497: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8498:                                     const CategoryManager: IUnknown = nil): HResult;
8499: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8500:                                      const CategoryManager: IUnknown = nil): HResult;
8501: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8502:                    const ADelete: Boolean = True): WideString;
8503: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8504: Function XRTLGetVariantAsString( const Value : Variant) : string
8505: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8506: Function XRTLGetTimeZones : TXRTLTimeZones
8507: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8508: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8509: Function GMTNow : TDateTime
8510: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8511: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8512: Procedure XRTLNotImplemented
8513: Procedure XRTLRaiseError( E : Exception)
8514: Procedure XRTLRaise( E : Exception)');
8515: Procedure XRaise( E : Exception)');
8516: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8517:
8518:
8519: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8520: begin
8521:   SIRegister_IXRTLValue(CL);
8522:   SIRegister_TXRTLValue(CL);
8523:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8524:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8525: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8526: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8527: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8528: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8529: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8530: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8531: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8532: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8533: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8534: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8535: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8536: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8537: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8538: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8539: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8540: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8541: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8542: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8543: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8544: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8545: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8546: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8547: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8548: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8549: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8550: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8551: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8552:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8553: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8554: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8555: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8556: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8557: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
```

```
8558: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8559: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8560: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8561: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8562: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8563: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8564: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8565: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8566: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8567: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8568: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8569: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8570: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8571: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8572: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8573: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8574: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8575: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8576: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8577: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8578: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8579: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8580: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8581: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8582: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8583: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8584: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8585: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8586: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8587: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8588: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8589: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8590: end;
8591:
8592: //****************************unit uPSI_GR32;*****************************************
8593:
8594:  Function Color32( WinColor : TColor) : TColor32;
8595:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8596:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8597:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8598:  Function WinColor( Color32 : TColor32) : TColor
8599:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8600:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8601:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8602:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8603:  Function RedComponent( Color32 : TColor32) : Integer
8604:  Function GreenComponent( Color32 : TColor32) : Integer
8605:  Function BlueComponent( Color32 : TColor32) : Integer
8606:  Function AlphaComponent( Color32 : TColor32) : Integer
8607:  Function Intensity( Color32 : TColor32) : Integer
8608:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8609:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8610:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8611:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8612:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8613:  Function WinPalette( const P : TPalette32) : HPALETTE
8614:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8615:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8616:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8617:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8618:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8619:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8620:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8621:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8622:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8623:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8624:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8625:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8626:  Function FixedRect1( const ARect : TRect) : TRect;
8627:  Function FixedRect2( const FR : TFloatRect) : TRect;
8628:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8629:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8630:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8631:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8632:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8633:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8634:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8635:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8636:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8637:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8638:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8639:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8640:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8641:  Function IsRectEmpty( const R : TRect) : Boolean;
8642:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8643:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8644:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8645:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
```

```
8646:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8647:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8648:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8649:  Function MessageBeep( uType : UINT) : BOOL
8650:  Function ShowCursor( bShow : BOOL) : Integer
8651:  Function SetCursorPos( X, Y : Integer) : BOOL
8652:  Function SetCursor( hCursor : HICON) : HCURSOR
8653:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8654:  //Function ClipCursor( lpRect : PRect) : BOOL
8655:  Function GetClipCursor( var lpRect : TRect) : BOOL
8656:  Function GetCursor : HCURSOR
8657:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8658:  Function GetCaretBlinkTime : UINT
8659:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8660:  Function DestroyCaret : BOOL
8661:  Function HideCaret( hWnd : HWND) : BOOL
8662:  Function ShowCaret( hWnd : HWND) : BOOL
8663:  Function SetCaretPos( X, Y : Integer) : BOOL
8664:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8665:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8666:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8667:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8668:  Function WindowFromPoint( Point : TPoint) : HWND
8669:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8670:
8671:
8672: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8673: begin
8674:  Function FixedFloor( A : TFixed) : Integer
8675:  Function FixedCeil( A : TFixed) : Integer
8676:  Function FixedMul( A, B : TFixed) : TFixed
8677:  Function FixedDiv( A, B : TFixed) : TFixed
8678:  Function OneOver( Value : TFixed) : TFixed
8679:  Function FixedRound( A : TFixed) : Integer
8680:  Function FixedSqr( Value : TFixed) : TFixed
8681:  Function FixedSqrtLP( Value : TFixed) : TFixed
8682:  Function FixedSqrtHP( Value : TFixed) : TFixed
8683:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8684:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8685:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8686:  Function GRHypot( const X, Y : TFloat) : TFloat;
8687:  Function Hypot1( const X, Y : Integer) : Integer;
8688:  Function FastSqrt( const Value : TFloat) : TFloat
8689:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8690:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8691:  Function FastInvSqrt( const Value : Single) : Single;
8692:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8693:  Function GRIsPowerOf2( Value : Integer) : Boolean
8694:  Function PrevPowerOf2( Value : Integer) : Integer
8695:  Function NextPowerOf2( Value : Integer) : Integer
8696:  Function Average( A, B : Integer) : Integer
8697:  Function GRSign( Value : Integer) : Integer
8698:  Function FloatMod( x, y : Double) : Double
8699: end;
8700:
8701: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8702: begin
8703:  Function Clamp( const Value : Integer) : Integer;
8704:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8705:  Function StackAlloc( Size : Integer) : Pointer
8706:  Procedure StackFree( P : Pointer)
8707:  Procedure Swap( var A, B : Pointer);
8708:  Procedure Swap1( var A, B : Integer);
8709:  Procedure Swap2( var A, B : TFixed);
8710:  Procedure Swap3( var A, B : TColor32);
8711:  Procedure TestSwap( var A, B : Integer);
8712:  Procedure TestSwap1( var A, B : TFixed);
8713:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8714:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8715:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8716:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8717:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8718:  Function GRMin( const A, B, C : Integer) : Integer;
8719:  Function GRMax( const A, B, C : Integer) : Integer;
8720:  Function Clamp( Value, Max : Integer) : Integer;
8721:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8722:  Function Wrap( Value, Max : Integer) : Integer;
8723:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8724:  Function Wrap3( Value, Max : Single) : Single;;
8725:  Function WrapPow2( Value, Max : Integer) : Integer;
8726:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8727:  Function Mirror( Value, Max : Integer) : Integer;
8728:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8729:  Function MirrorPow2( Value, Max : Integer) : Integer;
8730:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8731:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8732:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8733:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8734:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
```

```
8735:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8736:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8737:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8738:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8739:  Function Div255( Value : Cardinal) : Cardinal
8740:  Function SAR_4( Value : Integer) : Integer
8741:  Function SAR_8( Value : Integer) : Integer
8742:  Function SAR_9( Value : Integer) : Integer
8743:  Function SAR_11( Value : Integer) : Integer
8744:  Function SAR_12( Value : Integer) : Integer
8745:  Function SAR_13( Value : Integer) : Integer
8746:  Function SAR_14( Value : Integer) : Integer
8747:  Function SAR_15( Value : Integer) : Integer
8748:  Function SAR_16( Value : Integer) : Integer
8749:  Function ColorSwap( WinColor : TColor) : TColor32
8750: end;
8751:
8752: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8753: begin
8754:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8755:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8756:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8757:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8758:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8759:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8760:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8761:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8762:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8763:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8764:  Function CreateBitmask( Components : TColor32Components) : TColor32
8765:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8766:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8767:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8768: end;
8769:
8770:
8771: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8772: begin
8773:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8774:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8775:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8776:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8777:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8778:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8779:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8780:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8781:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8782:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8783:    //+'tedRangeBuffer; MoreData : Boolean; end
8784:  Function NtfsSetSparse( const FileName : string) : Boolean
8785:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8786:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8787:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8788:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8789:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8790:  Function NtfsGetSparse( const FileName : string) : Boolean
8791:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8792:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8793:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8794:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8795:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8796:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8797:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8798:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8799:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8800:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8801:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8802:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8803:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8804:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8805:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8806:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8807:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8808:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8809:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8810:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8811:   AddTypeS('TStreamIds', 'set of TStreamId
8812:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8813:    +': ___Pointer; StreamIds : TStreamIds; end
8814:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8815:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8816:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8817:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8818:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
```

```
8819:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8820:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8821:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8822:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
      List:TStrings):Bool;
8823:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8824:  Function JclAppInstances : TJclAppInstances;
8825:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8826:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8827:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8828:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8829:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8830:
8831:
8832: (*----------------------------------------------------------------------------*)
8833: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8834: begin
8835:   FindClass('TOBJECT'),'EJclGraphicsError
8836:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8837:   TDynPointArray', 'array of TPoint
8838:   TDynDynPointArrayArray', 'array of TDynPointArray
8839:   TPointF', 'record X : Single; Y : Single; end
8840:   TDynPointArrayF', 'array of TPointF
8841:   TDrawMode2', '( dmOpaque, dmBlend )
8842:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8843:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8844:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8845:   TMatrix3d', 'record array[0..2,0..2] of extended end
8846:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8847:   TScanLine', 'array of Integer
8848:   TScanLines', 'array of TScanLine
8849:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8850:   TGradientDirection', '( gdVertical, gdHorizontal )
8851:   TPolyFillMode', '( fmAlternate, fmWinding )
8852:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8853:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8854:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8855:   SIRegister_TJclDesktopCanvas(CL);
8856:   FindClass('TOBJECT'),'TJclRegion
8857:   SIRegister_TJclRegionInfo(CL);
8858:   SIRegister_TJclRegion(CL);
8859:   SIRegister_TJclThreadPersistent(CL);
8860:   SIRegister_TJclCustomMap(CL);
8861:   SIRegister_TJclBitmap32(CL);
8862:   SIRegister_TJclByteMap(CL);
8863:   SIRegister_TJclTransformation(CL);
8864:   SIRegister_TJclLinearTransformation(CL);
8865:  Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8866:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8867:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8868:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8869:  Procedure BitmapToJPeg( const FileName : string)
8870:  Procedure JPegToBitmap( const FileName : string)
8871:  Function ExtractIconCount( const FileName : string) : Integer
8872:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8873:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8874:  Procedure
      BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8875:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8876:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8877:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8878:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
      TGradientDirection) : Boolean;
8879:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8880:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8881:  Procedure ScreenShot1( bm : TBitmap);
8882:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8883:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8884:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8885:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8886:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8887:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8888:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8889:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8890:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8891:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8892:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8893:  Procedure Invert( Dst, Src : TJclBitmap32)
8894:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8895:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8896:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8897:  Procedure SetGamma( Gamma : Single)
8898: end;
8899:
8900: (*----------------------------------------------------------------------------*)
8901: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
```

```
8902: begin
8903:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8904:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8905:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8906:  Function LockedDec( var Target : Integer) : Integer
8907:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8908:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8909:  Function LockedExchangeDec( var Target : Integer) : Integer
8910:  Function LockedExchangeInc( var Target : Integer) : Integer
8911:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8912:  Function LockedInc( var Target : Integer) : Integer
8913:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8914:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8915:   SIRegister_TJclDispatcherObject(CL);
8916:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8917:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8918:   SIRegister_TJclCriticalSection(CL);
8919:   SIRegister_TJclCriticalSectionEx(CL);
8920:   SIRegister_TJclEvent(CL);
8921:   SIRegister_TJclWaitableTimer(CL);
8922:   SIRegister_TJclSemaphore(CL);
8923:   SIRegister_TJclMutex(CL);
8924:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8925:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8926:   SIRegister_TJclOptex(CL);
8927:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8928:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8929:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8930:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8931:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8932:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8933:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8934:   PMeteredSection', '^TMeteredSection // will not work
8935:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8936:   SIRegister_TJclMeteredSection(CL);
8937:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8938:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8939:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8940:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8941:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8942:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8943:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8944:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8945:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8946:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8947:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8948:   FindClass('TOBJECT'),'EJclCriticalSectionError
8949:   FindClass('TOBJECT'),'EJclEventError
8950:   FindClass('TOBJECT'),'EJclWaitableTimerError
8951:   FindClass('TOBJECT'),'EJclSemaphoreError
8952:   FindClass('TOBJECT'),'EJclMutexError
8953:   FindClass('TOBJECT'),'EJclMeteredSectionError
8954: end;
8955:
8956:
8957: //**************************unit uPSI_mORMotReport;
8958: Procedure SetCurrentPrinterAsDefault
8959: Function CurrentPrinterName : string
8960: Function mCurrentPrinterPaperSize : string
8961: Procedure UseDefaultPrinter
8962:
8963: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
8964: begin
8965:   with FindClass('TOBJECT'), 'TStream') do begin
8966:     IsAbstract := True;
8967:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8968:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8969:     function Read(Buffer:String;Count:LongInt):LongInt
8970:     function Write(Buffer:String;Count:LongInt):LongInt
8971:     function ReadString(Buffer:String;Count:LongInt):LongInt    //FileStream
8972:     function WriteString(Buffer:String;Count:LongInt):LongInt
8973:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
8974:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
8975:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8976:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8977:
8978:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8979:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8980:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8981:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8982:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8983:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8984:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8985:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8986:
8987:     function Seek(Offset:LongInt;Origin:Word):LongInt
8988:     procedure ReadBuffer(Buffer:String;Count:LongInt)
```

```
8989:     procedure WriteBuffer(Buffer:String;Count:LongInt)
8990:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
8991:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)');
8992:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)');
8993:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)');
8994:
8995:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
8996:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
8997:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
8998:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
8999:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9000:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9001:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9002:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9003:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9004:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9005:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9006:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9007:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9008:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9009:
9010:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
9011:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
9012:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
9013:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)');
9014:     //READBUFFERAC
9015:     function InstanceSize: Longint
9016:     Procedure FixupResourceHeader( FixupInfo : Integer)
9017:     Procedure ReadResHeader
9018:
9019:     {$IFDEF DELPHI4UP}
9020:     function CopyFrom(Source:TStream;Count:Int64):LongInt
9021:     {$ELSE}
9022:     function CopyFrom(Source:TStream;Count:Integer):LongInt
9023:     {$ENDIF}
9024:     RegisterProperty('Position', 'LongInt', iptrw);
9025:     RegisterProperty('Size', 'LongInt', iptrw);
9026:   end;
9027: end;
9028:
9029:
9030: { ********************************************************************
9031:   Unit DMATH - Interface for DMATH.DLL
9032:   ******************************************************************** }
9033: // see more docs/dmath_manual.pdf
9034:
9035: Function InitEval : Integer
9036: Procedure SetVariable( VarName : Char; Value : Float)
9037: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9038: Function Eval( ExpressionString : String) : Float
9039:
9040: unit dmath; //types are in built, others are external in DLL
9041: interface
9042: {$IFDEF DELPHI}
9043: uses
9044:   StdCtrls, Graphics;
9045: {$ENDIF}
9046: { ------------------------------------------------------------------
9047:   Types and constants
9048:   ------------------------------------------------------------------ }
9049: {$i types.inc}
9050: { ------------------------------------------------------------------
9051:   Error handling
9052:   ------------------------------------------------------------------ }
9053: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9054: { Sets the error code }
9055: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9056: { Sets error code and default function value }
9057: function MathErr : Integer; external 'dmath';
9058: { Returns the error code }
9059: { ------------------------------------------------------------------
9060:   Dynamic arrays
9061:   ------------------------------------------------------------------ }
9062: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9063: { Sets the auto-initialization of arrays }
9064: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9065: { Creates floating point vector V[0..Ub] }
9066: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9067: { Creates integer vector V[0..Ub] }
9068: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9069: { Creates complex vector V[0..Ub] }
9070: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9071: { Creates boolean vector V[0..Ub] }
9072: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9073: { Creates string vector V[0..Ub] }
9074: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9075: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9076: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9077: { Creates integer matrix A[0..Ub1, 0..Ub2] }
```

```
9078: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9079: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9080: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9081: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9082: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9083: { Creates string matrix A[0..Ub1, 0..Ub2] }
9084: { ------------------------------------------------------------------
9085:   Minimum, maximum, sign and exchange
9086:   ------------------------------------------------------------------ }
9087: function FMin(X, Y : Float) : Float; external 'dmath';
9088: { Minimum of 2 reals }
9089: function FMax(X, Y : Float) : Float; external 'dmath';
9090: { Maximum of 2 reals }
9091: function IMin(X, Y : Integer) : Integer; external 'dmath';
9092: { Minimum of 2 integers }
9093: function IMax(X, Y : Integer) : Integer; external 'dmath';
9094: { Maximum of 2 integers }
9095: function Sgn(X : Float) : Integer; external 'dmath';
9096: { Sign (returns 1 if X = 0) }
9097: function Sgn0(X : Float) : Integer; external 'dmath';
9098: { Sign (returns 0 if X = 0) }
9099: function DSgn(A, B : Float) : Float; external 'dmath';
9100: { Sgn(B) * |A| }
9101: procedure FSwap(var X, Y : Float); external 'dmath';
9102: { Exchange 2 reals }
9103: procedure ISwap(var X, Y : Integer); external 'dmath';
9104: { Exchange 2 integers }
9105: { ------------------------------------------------------------------
9106:   Rounding functions
9107:   ------------------------------------------------------------------ }
9108: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9109: { Rounds X to N decimal places }
9110: function Ceil(X : Float) : Integer; external 'dmath';
9111: { Ceiling function }
9112: function Floor(X : Float) : Integer; external 'dmath';
9113: { Floor function }
9114: { ------------------------------------------------------------------
9115:   Logarithms, exponentials and power
9116:   ------------------------------------------------------------------ }
9117: function Expo(X : Float) : Float; external 'dmath';
9118: { Exponential }
9119: function Exp2(X : Float) : Float; external 'dmath';
9120: { 2^X }
9121: function Exp10(X : Float) : Float; external 'dmath';
9122: { 10^X }
9123: function Log(X : Float) : Float; external 'dmath';
9124: { Natural log }
9125: function Log2(X : Float) : Float; external 'dmath';
9126: { Log, base 2 }
9127: function Log10(X : Float) : Float; external 'dmath';
9128: { Decimal log }
9129: function LogA(X, A : Float) : Float; external 'dmath';
9130: { Log, base A }
9131: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9132: { X^N }
9133: function Power(X, Y : Float) : Float; external 'dmath';
9134: { X^Y, X >= 0 }
9135: { ------------------------------------------------------------------
9136:   Trigonometric functions
9137:   ------------------------------------------------------------------ }
9138: function Pythag(X, Y : Float) : Float; external 'dmath';
9139: { Sqrt(X^2 + Y^2) }
9140: function FixAngle(Theta : Float) : Float; external 'dmath';
9141: { Set Theta in -Pi..Pi }
9142: function Tan(X : Float) : Float; external 'dmath';
9143: { Tangent }
9144: function ArcSin(X : Float) : Float; external 'dmath';
9145: { Arc sinus }
9146: function ArcCos(X : Float) : Float; external 'dmath';
9147: { Arc cosinus }
9148: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9149: { Angle (Ox, OM) with M(X,Y) }
9150: { ------------------------------------------------------------------
9151:   Hyperbolic functions
9152:   ------------------------------------------------------------------ }
9153: function Sinh(X : Float) : Float; external 'dmath';
9154: { Hyperbolic sine }
9155: function Cosh(X : Float) : Float; external 'dmath';
9156: { Hyperbolic cosine }
9157: function Tanh(X : Float) : Float; external 'dmath';
9158: { Hyperbolic tangent }
9159: function ArcSinh(X : Float) : Float; external 'dmath';
9160: { Inverse hyperbolic sine }
9161: function ArcCosh(X : Float) : Float; external 'dmath';
9162: { Inverse hyperbolic cosine }
9163: function ArcTanh(X : Float) : Float; external 'dmath';
9164: { Inverse hyperbolic tangent }
9165: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9166: { Sinh & Cosh }
```

```
9167: { ------------------------------------------------------------------
9168:   Gamma function and related functions
9169:   ------------------------------------------------------------------ }
9170: function Fact(N : Integer) : Float; external 'dmath';
9171: { Factorial }
9172: function SgnGamma(X : Float) : Integer; external 'dmath';
9173: { Sign of Gamma function }
9174: function Gamma(X : Float) : Float; external 'dmath';
9175: { Gamma function }
9176: function LnGamma(X : Float) : Float; external 'dmath';
9177: { Logarithm of Gamma function }
9178: function Stirling(X : Float) : Float; external 'dmath';
9179: { Stirling's formula for the Gamma function }
9180: function StirLog(X : Float) : Float; external 'dmath';
9181: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9182: function DiGamma(X : Float ) : Float; external 'dmath';
9183: { Digamma function }
9184: function TriGamma(X : Float ) : Float; external 'dmath';
9185: { Trigamma function }
9186: function IGamma(A, X : Float) : Float; external 'dmath';
9187: { Incomplete Gamma function}
9188: function JGamma(A, X : Float) : Float; external 'dmath';
9189: { Complement of incomplete Gamma function }
9190: function InvGamma(A, P : Float) : Float; external 'dmath';
9191: { Inverse of incomplete Gamma function }
9192: function Erf(X : Float) : Float; external 'dmath';
9193: { Error function }
9194: function Erfc(X : Float) : Float; external 'dmath';
9195: { Complement of error function }
9196: { ------------------------------------------------------------------
9197:   Beta function and related functions
9198:   ------------------------------------------------------------------ }
9199: function Beta(X, Y : Float) : Float; external 'dmath';
9200: { Beta function }
9201: function IBeta(A, B, X : Float) : Float; external 'dmath';
9202: { Incomplete Beta function }
9203: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9204: { Inverse of incomplete Beta function }
9205: { ------------------------------------------------------------------
9206:   Lambert's function
9207:   ------------------------------------------------------------------ }
9208: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9209:  ------------------------------------------------------------------
9210:   Binomial distribution
9211:   ------------------------------------------------------------------ }
9212: function Binomial(N, K : Integer) : Float; external 'dmath';
9213: { Binomial coefficient C(N,K) }
9214: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9215: { Probability of binomial distribution }
9216: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9217: { Cumulative probability for binomial distrib. }
9218: { ------------------------------------------------------------------
9219:   Poisson distribution
9220:   ------------------------------------------------------------------ }
9221: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9222: { Probability of Poisson distribution }
9223: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9224: { Cumulative probability for Poisson distrib. }
9225: { ------------------------------------------------------------------
9226:   Exponential distribution
9227:   ------------------------------------------------------------------ }
9228: function DExpo(A, X : Float) : Float; external 'dmath';
9229: { Density of exponential distribution with parameter A }
9230: function FExpo(A, X : Float) : Float; external 'dmath';
9231: { Cumulative probability function for exponential dist. with parameter A }
9232: { ------------------------------------------------------------------
9233:   Standard normal distribution
9234:   ------------------------------------------------------------------ }
9235: function DNorm(X : Float) : Float; external 'dmath';
9236: { Density of standard normal distribution }
9237: function FNorm(X : Float) : Float; external 'dmath';
9238: { Cumulative probability for standard normal distrib. }
9239: function PNorm(X : Float) : Float; external 'dmath';
9240: { Prob(|U| > X) for standard normal distrib. }
9241: function InvNorm(P : Float) : Float; external 'dmath';
9242: { Inverse of standard normal distribution }
9243: { ------------------------------------------------------------------
9244:   Student's distribution
9245:   ------------------------------------------------------------------ }
9246: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9247: { Density of Student distribution with Nu d.o.f. }
9248: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9249: { Cumulative probability for Student distrib. with Nu d.o.f. }
9250: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9251: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9252: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9253: { Inverse of Student's t-distribution function }
9254: { ------------------------------------------------------------------
9255:   Khi-2 distribution
```

```
9256:   ------------------------------------------------------------------ }
9257: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9258: { Density of Khi-2 distribution with Nu d.o.f. }
9259: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9260: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9261: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9262: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9263: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9264: { Inverse of Khi-2 distribution function }
9265: { ------------------------------------------------------------------
9266:   Fisher-Snedecor distribution
9267:   ------------------------------------------------------------------ }
9268: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9269: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9270: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9271: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9272: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9273: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9274: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9275: { Inverse of Snedecor's F-distribution function }
9276: { ------------------------------------------------------------------
9277:   Beta distribution
9278:   ------------------------------------------------------------------ }
9279: function DBeta(A, B, X : Float) : Float; external 'dmath';
9280: { Density of Beta distribution with parameters A and B }
9281: function FBeta(A, B, X : Float) : Float; external 'dmath';
9282: { Cumulative probability for Beta distrib. with param. A and B }
9283: { ------------------------------------------------------------------
9284:   Gamma distribution
9285:   ------------------------------------------------------------------ }
9286: function DGamma(A, B, X : Float) : Float; external 'dmath';
9287: { Density of Gamma distribution with parameters A and B }
9288: function FGamma(A, B, X : Float) : Float; external 'dmath';
9289: { Cumulative probability for Gamma distrib. with param. A and B }
9290: { ------------------------------------------------------------------
9291:   Expression evaluation
9292:   ------------------------------------------------------------------ }
9293: function InitEval : Integer; external 'dmath';
9294: { Initializes built-in functions and returns their number }
9295: function Eval(ExpressionString : String) : Float; external 'dmath';
9296: { Evaluates an expression at run-time }
9297: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9298: { Assigns a value to a variable }
9299: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9300: { Adds a function to the parser }
9301: { ------------------------------------------------------------------
9302:   Matrices and linear equations
9303:   ------------------------------------------------------------------ }
9304: procedure GaussJordan(A           : TMatrix;
9305:                       Lb, Ub1, Ub2 : Integer;
9306:                       var Det      : Float); external 'dmath';
9307: { Transforms a matrix according to the Gauss-Jordan method }
9308: procedure LinEq(A       : TMatrix;
9309:                 B       : TVector;
9310:                 Lb, Ub  : Integer;
9311:                 var Det : Float); external 'dmath';
9312: { Solves a linear system according to the Gauss-Jordan method }
9313: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9314: { Cholesky factorization of a positive definite symmetric matrix }
9315: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9316: { LU decomposition }
9317: procedure LU_Solve(A      : TMatrix;
9318:                    B      : TVector;
9319:                    Lb, Ub : Integer;
9320:                    X      : TVector); external 'dmath';
9321: { Solution of linear system from LU decomposition }
9322: procedure QR_Decomp(A           : TMatrix;
9323:                     Lb, Ub1, Ub2 : Integer;
9324:                     R           : TMatrix); external 'dmath';
9325: { QR decomposition }
9326: procedure QR_Solve(Q, R         : TMatrix;
9327:                    B           : TVector;
9328:                    Lb, Ub1, Ub2 : Integer;
9329:                    X           : TVector); external 'dmath';
9330: { Solution of linear system from QR decomposition }
9331: procedure SV_Decomp(A           : TMatrix;
9332:                     Lb, Ub1, Ub2 : Integer;
9333:                     S           : TVector;
9334:                     V           : TMatrix); external 'dmath';
9335: { Singular value decomposition }
9336: procedure SV_SetZero(S      : TVector;
9337:                      Lb, Ub : Integer;
9338:                      Tol    : Float); external 'dmath';
9339: { Set lowest singular values to zero }
9340: procedure SV_Solve(U           : TMatrix;
9341:                    S           : TVector;
9342:                    V           : TMatrix;
9343:                    B           : TVector;
9344:                    Lb, Ub1, Ub2 : Integer;
```

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

```
9434:                   MaxIter   : Integer;
9435:                   Tol       : Float;
9436:                   var F_min : Float); external 'dmath';
9437: { Minimization of a function of several variables (Simplex) }
9438: procedure SaveSimplex(FileName : string); external 'dmath';
9439: { Save Simplex iterations in a file }
9440: { ---------------------------------------------------------------
9441:   Nonlinear equations
9442:   --------------------------------------------------------------- }
9443: procedure RootBrack(Func            : TFunc;
9444:                     var X, Y, FX, FY : Float); external 'dmath';
9445: { Brackets a root of function Func between X and Y }
9446: procedure Bisect(Func     : TFunc;
9447:                  var X, Y : Float;
9448:                  MaxIter  : Integer;
9449:                  Tol      : Float;
9450:                  var F    : Float); external 'dmath';
9451: { Bisection method }
9452: procedure Secant(Func     : TFunc;
9453:                  var X, Y : Float;
9454:                  MaxIter  : Integer;
9455:                  Tol      : Float;
9456:                  var F    : Float); external 'dmath';
9457: { Secant method }
9458: procedure NewtEq(Func, Deriv : TFunc;
9459:                  var X       : Float;
9460:                  MaxIter     : Integer;
9461:                  Tol         : Float;
9462:                  var F       : Float); external 'dmath';
9463: { Newton-Raphson method for a single nonlinear equation }
9464: procedure NewtEqs(Equations : TEquations;
9465:                   Jacobian  : TJacobian;
9466:                   X, F      : TVector;
9467:                   Lb, Ub    : Integer;
9468:                   MaxIter   : Integer;
9469:                   Tol       : Float); external 'dmath';
9470: { Newton-Raphson method for a system of nonlinear equations }
9471: procedure Broyden(Equations : TEquations;
9472:                   X, F      : TVector;
9473:                   Lb, Ub    : Integer;
9474:                   MaxIter   : Integer;
9475:                   Tol       : Float); external 'dmath';
9476: { Broyden's method for a system of nonlinear equations }
9477: { ---------------------------------------------------------------
9478:   Polynomials and rational fractions
9479:   --------------------------------------------------------------- }
9480: function Poly(X    : Float;
9481:               Coef : TVector;
9482:               Deg  : Integer) : Float; external 'dmath';
9483: { Evaluates a polynomial }
9484: function RFrac(X          : Float;
9485:                Coef       : TVector;
9486:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9487: { Evaluates a rational fraction }
9488: function RootPol1(A, B  : Float;
9489:                   var X : Float) : Integer; external 'dmath';
9490: { Solves the linear equation A + B * X = 0 }
9491: function RootPol2(Coef : TVector;
9492:                   Z    : TCompVector) : Integer; external 'dmath';
9493: { Solves a quadratic equation }
9494: function RootPol3(Coef : TVector;
9495:                   Z    : TCompVector) : Integer; external 'dmath';
9496: { Solves a cubic equation }
9497: function RootPol4(Coef : TVector;
9498:                   Z    : TCompVector) : Integer; external 'dmath';
9499: { Solves a quartic equation }
9500: function RootPol(Coef : TVector;
9501:                  Deg  : Integer;
9502:                  Z    : TCompVector) : Integer; external 'dmath';
9503: { Solves a polynomial equation }
9504: function SetRealRoots(Deg : Integer;
9505:                       Z   : TCompVector;
9506:                       Tol : Float) : Integer; external 'dmath';
9507: { Set the imaginary part of a root to zero }
9508: procedure SortRoots(Deg : Integer;
9509:                     Z   : TCompVector); external 'dmath';
9510: { Sorts the roots of a polynomial }
9511: { ---------------------------------------------------------------
9512:   Numerical integration and differential equations
9513:   --------------------------------------------------------------- }
9514: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9515: { Integration by trapezoidal rule }
9516: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9517: { Integral from A to B }
9518: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9519: { Integral from 0 to B }
9520: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9521: { Convolution product at time T }
9522: procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
```

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

```
9612: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9613: { Initializes Simulated Annealing parameters }
9614: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9615: { Initializes log file }
9616: procedure SimAnn(Func          : TFuncNVar;
9617:                  X, Xmin, Xmax : TVector;
9618:                  Lb, Ub        : Integer;
9619:                  var F_min     : Float); external 'dmath';
9620: { Minimization of a function of several var. by simulated annealing }
9621: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9622: { Initializes Genetic Algorithm parameters }
9623: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9624: { Initializes log file }
9625: procedure GenAlg(Func          : TFuncNVar;
9626:                  X, Xmin, Xmax : TVector;
9627:                  Lb, Ub        : Integer;
9628:                  var F_min     : Float); external 'dmath';
9629: { Minimization of a function of several var. by genetic algorithm }
9630: { ---------------------------------------------------------------
9631:   Statistics
9632:   --------------------------------------------------------------- }
9633: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9634: { Mean of sample X }
9635: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9636: { Minimum of sample X }
9637: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9638: { Maximum of sample X }
9639: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9640: { Median of sample X }
9641: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9642: { Standard deviation estimated from sample X }
9643: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9644: { Standard deviation of population }
9645: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9646: { Correlation coefficient }
9647: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9648: { Skewness of sample X }
9649: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9650: { Kurtosis of sample X }
9651: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9652: { Quick sort (ascending order) }
9653: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9654: { Quick sort (descending order) }
9655: procedure Interval(X1, X2             : Float;
9656:                    MinDiv, MaxDiv     : Integer;
9657:                    var Min, Max, Step : Float); external 'dmath';
9658: { Determines an interval for a set of values }
9659: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9660:                     var XMin, XMax, XStep : Float); external 'dmath';
9661: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9662: procedure StudIndep(N1, N2         : Integer;
9663:                     M1, M2, S1, S2 : Float;
9664:                     var T          : Float;
9665:                     var DoF        : Integer); external 'dmath';
9666: { Student t-test for independent samples }
9667: procedure StudPaired(X, Y    : TVector;
9668:                      Lb, Ub  : Integer;
9669:                      var T   : Float;
9670:                      var DoF : Integer); external 'dmath';
9671: { Student t-test for paired samples }
9672: procedure AnOVa1(Ns               : Integer;
9673:                  N                : TIntVector;
9674:                  M, S             : TVector;
9675:                  var V_f, V_r, F  : Float;
9676:                  var DoF_f, DoF_r : Integer); external 'dmath';
9677: { One-way analysis of variance }
9678: procedure AnOVa2(NA, NB, Nobs : Integer;
9679:                  M, S         : TMatrix;
9680:                  V, F         : TVector;
9681:                  DoF          : TIntVector); external 'dmath';
9682: { Two-way analysis of variance }
9683: procedure Snedecor(N1, N2         : Integer;
9684:                    S1, S2         : Float;
9685:                    var F          : Float;
9686:                    var DoF1, DoF2 : Integer); external 'dmath';
9687: { Snedecor's F-test (comparison of two variances) }
9688: procedure Bartlett(Ns       : Integer;
9689:                    N        : TIntVector;
9690:                    S        : TVector;
9691:                    var Khi2 : Float;
9692:                    var DoF  : Integer); external 'dmath';
9693: { Bartlett's test (comparison of several variances) }
9694: procedure Mann_Whitney(N1, N2     : Integer;
9695:                        X1, X2     : TVector;
9696:                        var U, Eps : Float); external 'dmath';
9697: { Mann-Whitney test}
9698: procedure Wilcoxon(X, Y       : TVector;
9699:                    Lb, Ub     : Integer;
9700:                    var Ndiff  : Integer;
```

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

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

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

```
 9968:                   B       : TVector;
 9969:                   V       : TMatrix); external 'dmath';
 9970: { Unweighted fit of Michaelis equation }
 9971: procedure WMichFit(X, Y, S : TVector;
 9972:                    Lb, Ub  : Integer;
 9973:                    MaxIter : Integer;
 9974:                    Tol     : Float;
 9975:                    B       : TVector;
 9976:                    V       : TMatrix); external 'dmath';
 9977: { Weighted fit of Michaelis equation }
 9978: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9979: { Returns the value of the Michaelis equation at point X }
 9980: procedure MintFit(X, Y    : TVector;
 9981:                   Lb, Ub  : Integer;
 9982:                   MintVar : TMintVar;
 9983:                   Fit_S0  : Boolean;
 9984:                   MaxIter : Integer;
 9985:                   Tol     : Float;
 9986:                   B       : TVector;
 9987:                   V       : TMatrix); external 'dmath';
 9988: { Unweighted fit of the integrated Michaelis equation }
 9989: procedure WMintFit(X, Y, S : TVector;
 9990:                    Lb, Ub  : Integer;
 9991:                    MintVar : TMintVar;
 9992:                    Fit_S0  : Boolean;
 9993:                    MaxIter : Integer;
 9994:                    Tol     : Float;
 9995:                    B       : TVector;
 9996:                    V       : TMatrix); external 'dmath';
 9997: { Weighted fit of the integrated Michaelis equation }
 9998: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9999: { Returns the value of the integrated Michaelis equation at point X }
10000: procedure HillFit(X, Y    : TVector;
10001:                  Lb, Ub  : Integer;
10002:                  MaxIter : Integer;
10003:                  Tol     : Float;
10004:                  B       : TVector;
10005:                  V       : TMatrix); external 'dmath';
10006: { Unweighted fit of Hill equation }
10007: procedure WHillFit(X, Y, S : TVector;
10008:                    Lb, Ub  : Integer;
10009:                    MaxIter : Integer;
10010:                    Tol     : Float;
10011:                    B       : TVector;
10012:                    V       : TMatrix); external 'dmath';
10013: { Weighted fit of Hill equation }
10014: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10015: { Returns the value of the Hill equation at point X }
10016: procedure LogiFit(X, Y     : TVector;
10017:                   Lb, Ub   : Integer;
10018:                   ConsTerm : Boolean;
10019:                   General  : Boolean;
10020:                   MaxIter  : Integer;
10021:                   Tol      : Float;
10022:                   B        : TVector;
10023:                   V        : TMatrix); external 'dmath';
10024: { Unweighted fit of logistic function }
10025: procedure WLogiFit(X, Y, S  : TVector;
10026:                    Lb, Ub   : Integer;
10027:                    ConsTerm : Boolean;
10028:                    General  : Boolean;
10029:                    MaxIter  : Integer;
10030:                    Tol      : Float;
10031:                    B        : TVector;
10032:                    V        : TMatrix); external 'dmath';
10033: { Weighted fit of logistic function }
10034: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10035: { Returns the value of the logistic function at point X }
10036: procedure PKFit(X, Y    : TVector;
10037:                 Lb, Ub  : Integer;
10038:                 MaxIter : Integer;
10039:                 Tol     : Float;
10040:                 B       : TVector;
10041:                 V       : TMatrix); external 'dmath';
10042: { Unweighted fit of the acid-base titration curve }
10043: procedure WPKFit(X, Y, S : TVector;
10044:                  Lb, Ub  : Integer;
10045:                  MaxIter : Integer;
10046:                  Tol     : Float;
10047:                  B       : TVector;
10048:                  V       : TMatrix); external 'dmath';
10049: { Weighted fit of the acid-base titration curve }
10050: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10051: { Returns the value of the acid-base titration function at point X }
10052: procedure PowFit(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 power function }
10059: procedure WPowFit(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 power function }
10066:
10067: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10068: { Returns the value of the power function at point X }
10069: procedure GammaFit(X, Y    : TVector;
10070:                    Lb, Ub  : Integer;
10071:                    MaxIter : Integer;
10072:                    Tol     : Float;
10073:                    B       : TVector;
10074:                    V       : TMatrix); external 'dmath';
10075: { Unweighted fit of gamma distribution function }
10076: procedure WGammaFit(X, Y, S : TVector;
10077:                     Lb, Ub  : Integer;
10078:                     MaxIter : Integer;
10079:                     Tol     : Float;
10080:                     B       : TVector;
10081:                     V       : TMatrix); external 'dmath';
10082: { Weighted fit of gamma distribution function }
10083: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10084: { Returns the value of the gamma distribution function at point X }
10085: { ------------------------------------------------------------------
10086:   Principal component analysis
10087:   ------------------------------------------------------------------ }
10088: procedure VecMean(X           : TMatrix;
10089:                   Lb, Ub, Nvar : Integer;
10090:                   M           : TVector); external 'dmath';
10091: { Computes the mean vector M from matrix X }
10092: procedure VecSD(X           : TMatrix;
10093:                 Lb, Ub, Nvar : Integer;
10094:                 M, S         : TVector); external 'dmath';
10095: { Computes the vector of standard deviations S from matrix X }
10096: procedure MatVarCov(X           : TMatrix;
10097:                     Lb, Ub, Nvar : Integer;
10098:                     M           : TVector;
10099:                     V           : TMatrix); external 'dmath';
10100: { Computes the variance-covariance matrix V from matrix X }
10101: procedure MatCorrel(V    : TMatrix;
10102:                     Nvar : Integer;
10103:                     R    : TMatrix); external 'dmath';
10104: { Computes the correlation matrix R from the var-cov matrix V }
10105: procedure PCA(R      : TMatrix;
10106:               Nvar   : Integer;
10107:               Lambda : TVector;
10108:               C, Rc  : TMatrix); external 'dmath';
10109: { Performs a principal component analysis of the correlation matrix R }
10110: procedure ScaleVar(X           : TMatrix;
10111:                    Lb, Ub, Nvar : Integer;
10112:                    M, S         : TVector;
10113:                    Z           : TMatrix); external 'dmath';
10114: { Scales a set of variables by subtracting means and dividing by SD's }
10115: procedure PrinFac(Z           : TMatrix;
10116:                   Lb, Ub, Nvar : Integer;
10117:                   C, F         : TMatrix); external 'dmath';
10118: { Computes principal factors }
10119: { ------------------------------------------------------------------
10120:   Strings
10121:   ------------------------------------------------------------------ }
10122: function LTrim(S : String) : String; external 'dmath';
10123: { Removes leading blanks }
10124: function RTrim(S : String) : String; external 'dmath';
10125: { Removes trailing blanks }
10126: function Trim(S : String) : String; external 'dmath';
10127: { Removes leading and trailing blanks }
10128: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10129: { Returns a string made of character C repeated N times }
10130: function RFill(S : String; L : Byte) : String; external 'dmath';
10131: { Completes string S with trailing blanks for a total length L }
10132: function LFill(S : String; L : Byte) : String; external 'dmath';
10133: { Completes string S with leading blanks for a total length L }
10134: function CFill(S : String; L : Byte) : String; external 'dmath';
10135: { Centers string S on a total length L }
10136: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10137: { Replaces in string S all the occurences of C1 by C2 }
10138: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10139: { Extracts a field from a string }
10140: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10141: { Parses a string into its constitutive fields }
10142: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10143: { Sets the numeric format }
10144: function FloatStr(X : Float) : String; external 'dmath';
10145: { Converts a real to a string according to the numeric format }
```

```
10146: function IntStr(N : LongInt) : String; external 'dmath';
10147: { Converts an integer to a string }
10148: function CompStr(Z : Complex) : String; external 'dmath';
10149: { Converts a complex number to a string }
10150: {$IFDEF DELPHI}
10151: function StrDec(S : String) : String; external 'dmath';
10152: { Set decimal separator to the symbol defined in SysUtils }
10153: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10154: { Test if a string represents a number and returns it in X }
10155: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10156: { Reads a floating point number from an Edit control }
10157: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10158: { Writes a floating point number in a text file }
10159: {$ENDIF}
10160: { ---------------------------------------------------------------
10161:   BGI / Delphi graphics
10162:   --------------------------------------------------------------- }
10163: function InitGraphics
10164: {$IFDEF DELPHI}
10165: (Width, Height : Integer) : Boolean;
10166: {$ELSE}
10167: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10168: { Enters graphic mode }
10169: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10170:                     X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10171: { Sets the graphic window }
10172: procedure SetOxScale(Scale                : TScale;
10173:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10174: { Sets the scale on the Ox axis }
10175: procedure SetOyScale(Scale                : TScale;
10176:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10177: { Sets the scale on the Oy axis }
10178: procedure GetOxScale(var Scale                  : TScale;
10179:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10180: { Returns the scale on the Ox axis }
10181: procedure GetOyScale(var Scale                  : TScale;
10182:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10183: { Returns the scale on the Oy axis }
10184: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10185: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10186: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10187: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10188: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10189: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10190: {$IFNDEF DELPHI}
10191: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10192: { Sets the font for the main graph title }
10193: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10194: { Sets the font for the Ox axis (title and labels) }
10195: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10196: { Sets the font for the Oy axis (title and labels) }
10197: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10198: { Sets the font for the legends }
10199: procedure SetClipping(Clip : Boolean); external 'dmath';
10200: { Determines whether drawings are clipped at the current viewport
10201:   boundaries, according to the value of the Boolean parameter Clip }
10202: {$ENDIF}
10203: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10204: { Plots the horizontal axis }
10205: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10206: { Plots the vertical axis }
10207: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10208: { Plots a grid on the graph }
10209: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10210: { Writes the title of the graph }
10211: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10212: { Sets the maximum number of curves and re-initializes their parameters }
10213: procedure SetPointParam
10214: {$IFDEF DELPHI}
10215: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10216: {$ELSE}
10217: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10218: { Sets the point parameters for curve # CurvIndex }
10219: procedure SetLineParam
10220: {$IFDEF DELPHI}
10221: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10222: {$ELSE}
10223: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10224: { Sets the line parameters for curve # CurvIndex }
10225: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10226: { Sets the legend for curve # CurvIndex }
10227: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10228: { Sets the step for curve # CurvIndex }
10229: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10230: procedure GetPointParam
10231: {$IFDEF DELPHI}
10232: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10233: {$ELSE}
10234: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
```

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

```
10324: { Writes the legends for the plotted curves }
10325: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z  : TVector; F : TMatrix); external 'dmath';
10326: { Contour plot }
10327: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10328: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10329:
10330: //*****************************************************unit uPSI_SynPdf;
10331:  Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10332:  Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10333:  Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10334:  Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10335:  Function PdfRect1( const Box : TPdfBox) : TPdfRect;
10336:  Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10337:  //Function _GetCharCount( Text : PAnsiChar) : integer
10338:  //Procedure L2R( W : PWideChar; L : integer)
10339:  Function PdfCoord( MM : single) : integer
10340:  Function CurrentPrinterPaperSize : TPDFPaperSize
10341:  Function CurrentPrinterRes : TPoint
10342:  Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10343:  Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
10344:  Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10345:  Const('Usp10','String 'usp10.dll
10346:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10347:    'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10348:   AddTypeS('TScriptState_set', 'set of TScriptState_enum
10349: //*********************************************************************
10350:
10351: procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10352: begin
10353:  Procedure PMrandomize( I : word)
10354:  Function PMrandom : longint
10355:  Function Rrand : extended
10356:  Function Irand( N : word) : word
10357:  Function Brand( P : extended) : boolean
10358:  Function Nrand : extended
10359: end;
10360:
10361: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10362: begin
10363:   Function Endian( x : LongWord) : LongWord
10364:   Function Endian64( x : Int64) : Int64
10365:   Function spRol( x : LongWord; y : Byte) : LongWord
10366:   Function spRor( x : LongWord; y : Byte) : LongWord
10367:   Function Ror64( x : Int64; y : Byte) : Int64
10368: end;
10369:
10370: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10371: begin
10372:  Procedure ClearModules
10373:  Procedure ReadMapFile( Fname : string)
10374:  Function AddressInfo( Address : dword) : string
10375: end;
10376:
10377: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10378: begin
10379:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10380:    +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10381:    +'teByOther, tpExecuteByOther )
10382:   TTarPermissions', 'set of TTarPermission
10383:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10384:    +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10385:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10386:   TTarModes', 'set of TTarMode
10387:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10388:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10389:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10390:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10391:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10392:   SIRegister_TTarArchive(CL);
10393:   SIRegister_TTarWriter(CL);
10394:  Function PermissionString( Permissions : TTarPermissions) : STRING
10395:  Function ConvertFilename( Filename : STRING) : STRING
10396:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10397:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10398:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10399: end;
10400:
10401:
10402: //*************************************************unit uPSI_TlHelp32;
10403: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10404: begin
10405:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10406:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10407:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10408:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10409:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10410:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10411:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10412:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
```

```
10413:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10414:   AddTypeS('THeapList32', 'tagHEAPLIST32
10415:  Const('HF32_DEFAULT','LongInt'( 1);
10416:  Const('HF32_SHARED','LongInt'( 2);
10417:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10418:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10419:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10420:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10421:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10422:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10423:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10424:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10425:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10426:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10427:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10428:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10429:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10430:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10431:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10432:    +'aPri : Longint; dwFlags : DWORD; end
10433:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10434:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10435:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10436:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10437: end;
10438:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10439:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10440:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10441:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10442:  Const('EWX_LOGOFF','LongInt'( 0);
10443:  Const('EWX_SHUTDOWN','LongInt'( 1);
10444:  Const('EWX_REBOOT','LongInt'( 2);
10445:  Const('EWX_FORCE','LongInt'( 4);
10446:  Const('EWX_POWEROFF','LongInt'( 8);
10447:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10448:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10449:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10450:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10451:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10452:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10453:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10454:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10455:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10456:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10457:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10458:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10459:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10460:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10461:  Function GetDesktopWindow : HWND
10462:  Function GetParent( hWnd : HWND) : HWND
10463:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10464:  Function GetTopWindow( hWnd : HWND) : HWND
10465:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10466:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10467:  //Delphi DFM
10468:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10469:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10470:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10471:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10472:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10473:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10474:  Function OpenIcon( hWnd : HWND) : BOOL
10475:  Function CloseWindow( hWnd : HWND) : BOOL
10476:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10477:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10478:  Function IsWindowVisible( hWnd : HWND) : BOOL
10479:  Function IsIconic( hWnd : HWND) : BOOL
10480:  Function AnyPopup : BOOL
10481:  Function BringWindowToTop( hWnd : HWND) : BOOL
10482:  Function IsZoomed( hWnd : HWND) : BOOL
10483:  Function IsWindow( hWnd : HWND) : BOOL
10484:  Function IsMenu( hMenu : HMENU) : BOOL
10485:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10486:  Function DestroyWindow( hWnd : HWND) : BOOL
10487:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10488:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10489:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10490:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10491:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10492:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10493:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10494:
10495: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10496: begin
10497:  const('ShowSetupDialogOptLong','String '--setup
10498:  PrimaryConfPathOptLong','String '--primary-config-path=
10499:  PrimaryConfPathOptShort','String '--pcp=
10500:  SecondaryConfPathOptLong','String '--secondary-config-path=
10501:  SecondaryConfPathOptShort','String '--scp=
```

```
10502:  NoSplashScreenOptLong','String '--no-splash-screen
10503:  NoSplashScreenOptShort','String '--nsc
10504:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10505:  SkipLastProjectOpt','String '--skip-last-project
10506:  DebugLogOpt','String '--debug-log=
10507:  DebugLogOptEnable','String '--debug-enable=
10508:  LanguageOpt','String '--language=
10509:  LazarusDirOpt','String '--lazarusdir=
10510:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10511:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10512:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10513:  Function IsHelpRequested : Boolean
10514:  Function IsVersionRequested : boolean
10515:  Function GetLanguageSpecified : string
10516:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10517:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10518:  Procedure ParseNoGuiCmdLineParams
10519:  Function ExtractCmdLineFilenames : TStrings
10520: end;
10521:
10522:
10523: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10524: begin
10525:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10526:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10527:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10528:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10529:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10530:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10531:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10532:  Function DirPathExists( DirectoryName : string) : boolean
10533:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10534:  Function ExtractFileNameOnly( const AFilename : string) : string
10535:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10536:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10537:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10538:  Function ForceDirectory( DirectoryName : string) : boolean
10539:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10540:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10541:  Function FileIsText( const AFilename : string) : boolean
10542:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10543:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10544:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10545:  Function TrimFilename( const AFilename : string) : string
10546:  Function ResolveDots( const AFilename : string) : string
10547:  Procedure ForcePathDelims( var FileName : string)
10548:  Function GetForcedPathDelims( const FileName : string) : String
10549:  Function CleanAndExpandFilename( const Filename : string) : string
10550:  Function CleanAndExpandDirectory( const Filename : string) : string
10551:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10552:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10553:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10554:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
       AlwaysRequireSharedBaseFolder: Boolean) : string
10555:  Function FileIsInPath( const Filename, Path : string) : boolean
10556:  Function AppendPathDelim( const Path : string) : string
10557:  Function ChompPathDelim( const Path : string) : string
10558:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10559:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10560:  Function MinimizeSearchPath( const SearchPath : string) : string
10561:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10562:  (*Function FileExistsUTF8( const Filename : string) : boolean
10563:  Function FileAgeUTF8( const FileName : string) : Longint
10564:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10565:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10566:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10567:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10568:  Procedure FindCloseUTF8( var F : TSearchrec)
10569:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10570:  Function FileGetAttrUTF8( const FileName : String) : Longint
10571:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10572:  Function DeleteFileUTF8( const FileName : String) : Boolean
10573:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10574:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10575:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10576:  Function GetCurrentDirUTF8 : String
10577:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10578:  Function CreateDirUTF8( const NewDir : String) : Boolean
10579:  Function RemoveDirUTF8( const Dir : String) : Boolean
10580:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10581:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10582:  Function FileCreateUTF8( const FileName : string) : THandle;
10583:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10584:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10585:  Function FileSizeUtf8( const Filename : string) : int64
10586:  Function GetFileDescription( const AFilename : string) : string
10587:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10588:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
```

```
10589:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10590:  Function IsUNCPath( const Path : String) : Boolean
10591:  Function ExtractUNCVolume( const Path : String) : String
10592:  Function ExtractFileRoot( FileName : String) : String
10593:  Function GetDarwinSystemFilename( Filename : string) : string
10594:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10595:  Function StrToCmdLineParam( const Param : string) : string
10596:  Function MergeCmdLineParams( ParamList : TStrings) : string
10597:  Procedure InvalidateFileStateCache( const Filename : string)
10598:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10599:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10600:  Function FindAllDocs(const Root, extmask: string): TStringlist;
10601:  Function ReadFileToString( const Filename : string) : string
10602:  procedure Inc1(var X: longint; N: Longint);
10603:
10604:  type
10605:   TCopyFileFlag = ( cffOverwriteFile,
10606:     cffCreateDestDirectory, cffPreserveTime );
10607:   TCopyFileFlags = set of TCopyFileFlag;*)
10608:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10609:   TCopyFileFlags', 'set of TCopyFileFlag
10610:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10611: end;
10612:
10613: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10614: begin
10615:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10616:   SIRegister_TMask(CL);
10617:   SIRegister_TParseStringList(CL);
10618:   SIRegister_TMaskList(CL);
10619:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10620:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10621:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10622:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10623: end;
10624:
10625: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10626: begin
10627:   //PShellHookInfo', '^TShellHookInfo // will not work
10628:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10629:   SHELLHOOKINFO', 'TShellHookInfo
10630:   LPSHELLHOOKINFO', 'PShellHookInfo
10631:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10632:   SIRegister_TJvShellHook(CL);
10633:  Function InitJvShellHooks : Boolean
10634:  Procedure UnInitJvShellHooks
10635: end;
10636:
10637: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10638: begin
10639:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10640:    +', dcHasSetSel, dcWantTab, dcNative )
10641:   TDlgCodes', 'set of TDlgCode
10642:  'dcWantMessage',' dcWantAllKeys);
10643:   SIRegister_IJvExControl(CL);
10644:   SIRegister_IJvDenySubClassing(CL);
10645:   SIRegister_TStructPtrMessage(CL);
10646:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10647:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10648:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10649:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10650:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10651:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10652:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10653:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10654:  Function GetFocusedControl( AControl : TControl) : TWinControl
10655:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10656:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10657:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10658:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10659:   SIRegister_TJvExControl(CL);
10660:   SIRegister_TJvExWinControl(CL);
10661:   SIRegister_TJvExCustomControl(CL);
10662:   SIRegister_TJvExGraphicControl(CL);
10663:   SIRegister_TJvExHintWindow(CL);
10664:   SIRegister_TJvExPubGraphicControl(CL);
10665: end;
10666:
10667: (*----------------------------------------------------------------------------*)
10668: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10669: begin
10670:  Procedure EncodeStream( Input, Output : TStream)
10671:  Procedure DecodeStream( Input, Output : TStream)
10672:  Function EncodeString1( const Input : string) : string
10673:  Function DecodeString1( const Input : string) : string
10674: end;
10675:
10676: (*----------------------------------------------------------------------------*)
10677: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
```

```
10678: begin
10679:   SIRegister_TWebAppRegInfo(CL);
10680:   SIRegister_TWebAppRegList(CL);
10681:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10682:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10683:  Procedure UnregisterWebApp( const AProgID : string)
10684:  Function FindRegisteredWebApp( const AProgID : string) : string
10685:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10686:  'sUDPPort','String 'UDPPort
10687: end;
10688:
10689: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10690: begin
10691:  // TStringDynArray', 'array of string
10692:  Function GetEnvVarValue( const VarName : string) : string
10693:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10694:  Function DeleteEnvVar( const VarName : string) : Integer
10695:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
      BufSize:Int):Int;
10696:  Function ExpandEnvVars( const Str : string) : string
10697:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10698:  Procedure GetAllEnvVarNames( const Names : TStrings);
10699:  Function GetAllEnvVarNames1 : TStringDynArray;
10700:  Function EnvBlockSize : Integer
10701:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10702:   SIRegister_TPJEnvVarsEnumerator(CL);
10703:   SIRegister_TPJEnvVars(CL);
10704:   FindClass('TOBJECT'),'EPJEnvVars
10705:   FindClass('TOBJECT'),'EPJEnvVars
10706:  //Procedure Register
10707: end;
10708:
10709: (*----------------------------------------------------------------------------*)
10710: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10711: begin
10712:  'cOneSecInMS','LongInt'( 1000);
10713:  //'cDefTimeSlice','LongInt'( 50);
10714:  //'cDefMaxExecTime',' cOneMinInMS);
10715:  'cAppErrorMask','LongInt'( 1 shl 29);
10716:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10717:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10718:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10719:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10720:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10721:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10722:  Function MakeSize( const ACX, ACY : LongInt) : TSize
10723:   SIRegister_TPJCustomConsoleApp(CL);
10724:   SIRegister_TPJConsoleApp(CL);
10725: end;
10726:
10727: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10728: begin
10729:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10730:   t_encoding', '( uuencode, base64, mime )
10731:  Function internet_date( date : TDateTime) : string
10732:  Function lookup_hostname( const hostname : string) : longint
10733:  Function my_hostname : string
10734:  Function my_ip_address : longint
10735:  Function ip2string( ip_address : longint) : string
10736:  Function resolve_hostname( ip : longint) : string
10737:  Function address_from( const s : string; count : integer) : string
10738:  Function encode_base64( data : TStream) : TStringList
10739:  Function decode_base64( source : TStringList) : TMemoryStream
10740:  Function posn( const s, t : string; count : integer) : integer
10741:  Function poscn( c : char; const s : string; n : integer) : integer
10742:  Function filename_of( const s : string) : string
10743:  //Function trim( const s : string) : string
10744:  //Procedure setlength( var s : string; l : byte)
10745:  Function TimeZoneBias : longint
10746:  Function eight2seven_quoteprint( const s : string) : string
10747:  Function eight2seven_german( const s : string) : string
10748:  Function seven2eight_quoteprint( const s : string) : string end;
10749:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10750:  Function socketerror : cint
10751:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10752:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10753:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10754:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10755:  Function fplisten( s : cint; backlog : cint) : cint
10756:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10757:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10758:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10759:  Function NetAddrToStr( Entry : in_addr) : String
10760:  Function HostAddrToStr( Entry : in_addr) : String
10761:  Function StrToHostAddr( IP : String) : in_addr
10762:  Function StrToNetAddr( IP : String) : in_addr
10763:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10764:   cint8', 'shortint
10765:   cuint8', 'byte
```

```
10766:   cchar', 'cint8
10767:   cschar', 'cint8
10768:   cuchar', 'cuint8
10769:   cint16', 'smallint
10770:   cuint16', 'word
10771:   cshort', 'cint16
10772:   csshort', 'cint16
10773:   cushort', 'cuint16
10774:   cint32', 'longint
10775:   cuint32', 'longword
10776:   cint', 'cint32
10777:   csint', 'cint32
10778:   cuint', 'cuint32
10779:   csigned', 'cint
10780:   cunsigned', 'cuint
10781:   cint64', 'int64
10782:   clonglong', 'cint64
10783:   cslonglong', 'cint64
10784:   cbool', 'longbool
10785:   cfloat', 'single
10786:   cdouble', 'double
10787:   clongdouble', 'extended
10788:
10789: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10790: begin
10791:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10792:   SIRegister_TlkJSONdotnetclass(CL);
10793:   SIRegister_TlkJSONbase(CL);
10794:   SIRegister_TlkJSONnumber(CL);
10795:   SIRegister_TlkJSONstring(CL);
10796:   SIRegister_TlkJSONboolean(CL);
10797:   SIRegister_TlkJSONnull(CL);
10798:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10799:    +'se; data : TObject; var Continue : Boolean)
10800:   SIRegister_TlkJSONcustomlist(CL);
10801:   SIRegister_TlkJSONlist(CL);
10802:   SIRegister_TlkJSONobjectmethod(CL);
10803:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10804:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10805:   SIRegister_TlkHashTable(CL);
10806:   SIRegister_TlkBalTree(CL);
10807:   SIRegister_TlkJSONobject(CL);
10808:   SIRegister_TlkJSON(CL);
10809:   SIRegister_TlkJSONstreamed(CL);
10810:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10811: end;
10812:
10813: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10814: begin
10815:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10816:   SIRegister_TZSortedList(CL);
10817:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10818:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10819:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10820:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10821:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10822:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10823:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10824:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10825:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10826:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10827:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10828:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10829:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10830:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10831:  Function StrToBoolEx( Str : string) : Boolean
10832:  Function BoolToStrEx( Bool : Boolean) : String
10833:  Function IsIpAddr( const Str : string) : Boolean
10834:  Function zSplitString( const Str, Delimiters : string) : TStrings
10835:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10836:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10837:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10838:  Function FloatToSQLStr( Value : Extended) : string
10839:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10840:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10841:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10842:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10843:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10844:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10845:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10846:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10847:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10848:  Function BytesToVar( const Value : TByteDynArray) : Variant
10849:  Function VarToBytes( const Value : Variant) : TByteDynArray
10850:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10851:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10852:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10853:  Function EncodeCString( const Value : string) : string
10854:  Function DecodeCString( const Value : string) : string
```

```
10855:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10856:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10857:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
        SubVersion:Int);
10858:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
        SubVersion:Integer):Int;
10859:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10860:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10861:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10862:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10863:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10864:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10865:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10866:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10867:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10868:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10869: end;
10870:
10871: unit uPSI_ZEncoding;
10872:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10873:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10874:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10875:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10876:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10877:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10878:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10879:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10880:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10881:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10882:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10883:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10884:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10885:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10886:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10887:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10888:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10889:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10890:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10891:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10892:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10893:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10894:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10895:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10896:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10897:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10898:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10899:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10900:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10901:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10902:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10903:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10904:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10905:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10906:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10907:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10908:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10909:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10910:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10911:  Function ZDefaultSystemCodePage : Word
10912:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10913:
10914:
10915: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10916: begin
10917:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10918:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10919:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10920:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10921:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10922:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10923:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10924:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
10925:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
10926:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
10927:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
10928:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
10929:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10930:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10931:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10932:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10933:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10934:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10935:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10936:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
10937:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
10938:  ('ilIdentiry','2 RPC_C_IMP_LEVEL_IDENTIFY);
10939:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
10940:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
10941:  ('EOAC_NONE','LongWord').SetUInt( $0);
```

```
10942:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10943:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10944:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10945:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10946:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10947:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10948:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10949:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10950:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10951:   FindClass('TOBJECT'),'EBoldCom
10952:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10953:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10954:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10955:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10956:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10957:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10958:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10959:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10960:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10961:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10962:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10963:  Function BoldCreateGUID : TGUID
10964:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
10965:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
10966:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10967:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10968: end;
10969:
10970: (*----------------------------------------------------------------------------*)
10971: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10972: begin
10973:  Function ParseISODate( s : string) : TDateTime
10974:  Function ParseISODateTime( s : string) : TDateTime
10975:  Function ParseISOTime( str : string) : TDateTime
10976: end;
10977:
10978: (*----------------------------------------------------------------------------*)
10979: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10980: begin
10981:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10982:  Function BoldCreateGUIDWithBracketsAsString : string
10983: end;
10984:
10985: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10986: begin
10987:   FindClass('TOBJECT'),'TBoldFileHandler
10988:   FindClass('TOBJECT'),'TBoldDiskFileHandler
10989:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
10990:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
10991:   SIRegister_TBoldFileHandler(CL);
10992:   SIRegister_TBoldDiskFileHandler(CL);
10993:  Procedure BoldCloseAllFilehandlers
10994:  Procedure BoldRemoveUnchangedFilesFromEditor
10995:  Function BoldFileHandlerList : TBoldObjectArray
10996:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
       OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
10997: end;
10998:
10999: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
11000: begin
11001:   PCharArr', 'array of PChar
11002:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
11003:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11004:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
11005:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
11006:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
11007:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
11008:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
11009:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
11010:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11011:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11012:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
       Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11013:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11014: end;
11015:
11016: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11017: begin
11018:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11019:   SIRegister_TfrmBoldQueryUser(CL);
11020:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
11021: end;
11022:
```

```
11023: (*----------------------------------------------------------------------------*)
11024: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
11025: begin
11026:  //('befIsInDisplayList',' BoldElementFlag0);
11027:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11028:  //('befFollowerSelected',' BoldElementFlag2);
11029:   FindClass('TOBJECT'),'TBoldQueue
11030:   FindClass('TOBJECT'),'TBoldQueueable
11031:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11032:   SIRegister_TBoldQueueable(CL);
11033:   SIRegister_TBoldQueue(CL);
11034:  Function BoldQueueFinalized : Boolean
11035:  Function BoldInstalledQueue : TBoldQueue
11036: end;
11037:
11038: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
11039: begin
11040:  const mmPerInch','Extended').setExtended( 25.4);
11041:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11042:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11043:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11044:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11045:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11046:   TBarLineType', '( white, black, black_half )
11047:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11048:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11049:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11050:   TCheckSumMethod', '( csmNone, csmModulo10 )
11051:   SIRegister_TAsBarcode(CL);
11052:  Function CheckSumModulo10( const data : string) : string
11053:  Function ConvertMmToPixelsX( const Value : Double) : Integer
11054:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11055:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11056:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11057: end;
11058:
11059: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11060: begin
11061:    THomogeneousByteVector', 'array[0..3] of Byte
11062:   THomogeneousWordVector', 'array[0..3] of Word
11063:   THomogeneousIntVector', 'array[0..3] of Integer
11064:   THomogeneousFltVector', 'array[0..3] of single
11065:   THomogeneousDblVector', 'array[0..3] of double
11066:   THomogeneousExtVector', 'array[0..3] of extended
11067:   TAffineByteVector', 'array[0..2] of Byte
11068:   TAffineWordVector', 'array[0..2] of Word
11069:   TAffineIntVector', 'array[0..2] of Integer
11070:   TAffineFltVector', 'array[0..2] of single
11071:   TAffineDblVector', 'array[0..2] of double
11072:   TAffineExtVector', 'array[0..2] of extended
11073:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11074:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11075:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11076:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11077:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11078:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11079:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11080:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11081:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11082:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11083:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11084:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11085:   TMatrix4b', 'THomogeneousByteMatrix
11086:   TMatrix4w', 'THomogeneousWordMatrix
11087:   TMatrix4i', 'THomogeneousIntMatrix
11088:   TMatrix4f', 'THomogeneousFltMatrix
11089:   TMatrix4d', 'THomogeneousDblMatrix
11090:   TMatrix4e', 'THomogeneousExtMatrix
11091:   TMatrix3b', 'TAffineByteMatrix
11092:   TMatrix3w', 'TAffineWordMatrix
11093:   TMatrix3i', 'TAffineIntMatrix
11094:   TMatrix3f', 'TAffineFltMatrix
11095:   TMatrix3d', 'TAffineDblMatrix
11096:   TMatrix3e', 'TAffineExtMatrix
11097:   //'PMatrix', '^TMatrix // will not work
11098:   TMatrixGL', 'THomogeneousFltMatrix
11099:   THomogeneousMatrix', 'THomogeneousFltMatrix
11100:   TAffineMatrix', 'TAffineFltMatrix
11101:   TQuaternion', 'record Vector : TVector4f; end
11102:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11103:   +'ger; Height : Integer; end
11104:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11105:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11106:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11107:  'EPSILON','Extended').setExtended( 1E-100);
11108:  'EPSILON2','Extended').setExtended( 1E-50);
11109:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11110:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11111:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
```

```
11112:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11113:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11114:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11115:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11116:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11117:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11118:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11119:  Function VectorLength( V : array of Single) : Single
11120:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11121:  Procedure VectorNegate( V : array of Single)
11122:  Function VectorNorm( V : array of Single) : Single
11123:  Function VectorNormalize( V : array of Single) : Single
11124:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11125:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11126:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11127:  Procedure VectorScale( V : array of Single; Factor : Single)
11128:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11129:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11130:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11131:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11132:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11133:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11134:  Procedure MatrixAdjoint( var M : TMatrixGL)
11135:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11136:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11137:  Function MatrixDeterminant( M : TMatrixGL) : Single
11138:  Procedure MatrixInvert( var M : TMatrixGL)
11139:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11140:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11141:  Procedure MatrixTranspose( var M : TMatrixGL)
11142:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11143:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11144:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11145:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11146:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11147:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11148:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11149:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11150:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11151:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11152:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11153:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11154:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11155:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11156:  Function MakeAffineVector( V : array of Single) : TAffineVector
11157:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11158:  Function MakeVector( V : array of Single) : TVectorGL
11159:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11160:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11161:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11162:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11163:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11164:  Function ArcCosGL( X : Extended) : Extended
11165:  Function ArcSinGL( X : Extended) : Extended
11166:  Function ArcTan2GL( Y, X : Extended) : Extended
11167:  Function CoTanGL( X : Extended) : Extended
11168:  Function DegToRadGL( Degrees : Extended) : Extended
11169:  Function RadToDegGL( Radians : Extended) : Extended
11170:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11171:  Function TanGL( X : Extended) : Extended
11172:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11173:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11174:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11175:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11176:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11177:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11178: end;
11179:
11180:
11181: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11182: begin
11183:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11184:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11185:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11186:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11187:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11188:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11189:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11190:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11191:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11192:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11193:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11194:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11195:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11196:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11197:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11198:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11199:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11200:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
```

```
11201:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11202:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11203:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11204:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11205:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11206:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11207:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
       Items:TStrings):Bool;
11208:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
       SaveTo:TStrings):Bool;
11209:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11210: end;
11211:
11212: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11213: begin
11214:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11215:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11216:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11217:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11218:   FindClass('TOBJECT'),'EInvalidParam
11219:  Function IsDCOMInstalled : Boolean
11220:  Function IsDCOMEnabled : Boolean
11221:  Function GetDCOMVersion : string
11222:  Function GetMDACVersion : string
11223:  Function GetMDACVersion2 : string
11224:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11225:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11226:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11227:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11228:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11229:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11230:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11231:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11232:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11233:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11234:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11235:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11236:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11237:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11238: end;
11239:
11240:
11241: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11242: begin
11243:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11244:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11245:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11246:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11247:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11248:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11249:  DegPerCycle','Extended').setExtended( 360.0);
11250:  DegPerGrad','Extended').setExtended( 0.9);
11251:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11252:  GradPerCycle','Extended').setExtended( 400.0);
11253:  GradPerDeg','Extended').setExtended( 1.111111111111111111111111111111);
11254:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11255:  RadPerCycle','Extended').setExtended( 6.283185307179586476925286766559);
11256:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11257:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11258:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11259:  CyclePerGrad','Extended').setExtended( 0.0025);
11260:  CyclePerRad','Extended').setExtended( 0.159154943091895335768883763372 51);
11261:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11262:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11263:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11264:  Function MakePercentage( const Step, Max : Longint) : Longint
11265:  Function CelsiusToKelvin( const T : double) : double
11266:  Function CelsiusToFahrenheit( const T : double) : double
11267:  Function KelvinToCelsius( const T : double) : double
11268:  Function KelvinToFahrenheit( const T : double) : double
11269:  Function FahrenheitToCelsius( const T : double) : double
11270:  Function FahrenheitToKelvin( const T : double) : double
11271:  Function CycleToDeg( const Cycles : double) : double
11272:  Function CycleToGrad( const Cycles : double) : double
11273:  Function CycleToRad( const Cycles : double) : double
11274:  Function DegToCycle( const Degrees : double) : double
11275:  Function DegToGrad( const Degrees : double) : double
11276:  Function DegToRad( const Degrees : double) : double
11277:  Function GradToCycle( const Grads : double) : double
11278:  Function GradToDeg( const Grads : double) : double
11279:  Function GradToRad( const Grads : double) : double
11280:  Function RadToCycle( const Radians : double) : double
11281:  Function RadToDeg( const Radians : double) : double
11282:  Function RadToGrad( const Radians : double) : double
11283:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11284:  Function DmsToRad( const D, M : Integer; const S : double) : double
```

```
11285:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11286:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11287:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11288:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11289:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11290:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11291:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11292:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11293:  Function CmToInch( const Cm : double) : double
11294:  Function InchToCm( const Inch : double) : double
11295:  Function FeetToMetre( const Feet : double) : double
11296:  Function MetreToFeet( const Metre : double) : double
11297:  Function YardToMetre( const Yard : double) : double
11298:  Function MetreToYard( const Metre : double) : double
11299:  Function NmToKm( const Nm : double) : double
11300:  Function KmToNm( const Km : double) : double
11301:  Function KmToSm( const Km : double) : double
11302:  Function SmToKm( const Sm : double) : double
11303:  Function LitreToGalUs( const Litre : double) : double
11304:  Function GalUsToLitre( const GalUs : double) : double
11305:  Function GalUsToGalCan( const GalUs : double) : double
11306:  Function GalCanToGalUs( const GalCan : double) : double
11307:  Function GalUsToGalUk( const GalUs : double) : double
11308:  Function GalUkToGalUs( const GalUk : double) : double
11309:  Function LitreToGalCan( const Litre : double) : double
11310:  Function GalCanToLitre( const GalCan : double) : double
11311:  Function LitreToGalUk( const Litre : double) : double
11312:  Function GalUkToLitre( const GalUk : double) : double
11313:  Function KgToLb( const Kg : double) : double
11314:  Function LbToKg( const Lb : double) : double
11315:  Function KgToOz( const Kg : double) : double
11316:  Function OzToKg( const Oz : double) : double
11317:  Function CwtUsToKg( const Cwt : double) : double
11318:  Function CwtUkToKg( const Cwt : double) : double
11319:  Function KaratToKg( const Karat : double) : double
11320:  Function KgToCwtUs( const Kg : double) : double
11321:  Function KgToCwtUk( const Kg : double) : double
11322:  Function KgToKarat( const Kg : double) : double
11323:  Function KgToSton( const Kg : double) : double
11324:  Function KgToLton( const Kg : double) : double
11325:  Function StonToKg( const STon : double) : double
11326:  Function LtonToKg( const Lton : double) : double
11327:  Function QrUsToKg( const Qr : double) : double
11328:  Function QrUkToKg( const Qr : double) : double
11329:  Function KgToQrUs( const Kg : double) : double
11330:  Function KgToQrUk( const Kg : double) : double
11331:  Function PascalToBar( const Pa : double) : double
11332:  Function PascalToAt( const Pa : double) : double
11333:  Function PascalToTorr( const Pa : double) : double
11334:  Function BarToPascal( const Bar : double) : double
11335:  Function AtToPascal( const At : double) : double
11336:  Function TorrToPascal( const Torr : double) : double
11337:  Function KnotToMs( const Knot : double) : double
11338:  Function HpElectricToWatt( const HpE : double) : double
11339:  Function HpMetricToWatt( const HpM : double) : double
11340:  Function MsToKnot( const ms : double) : double
11341:  Function WattToHpElectric( const W : double) : double
11342:  Function WattToHpMetric( const W : double) : double
11343:  function getBigPI: string;    //PI of 1000 numbers
11344:
11345: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11346: begin
11347:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11348:  Procedure CDCopyFile( const FileName, DestName : string)
11349:  Procedure CDMoveFile( const FileName, DestName : string)
11350:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11351:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11352:  Function CDGetTempDir : string
11353:  Function CDGetFileSize( FileName : string) : longint
11354:  Function GetFileTime( FileName : string) : longint
11355:  Function GetShortName( FileName : string) : string
11356:  Function GetFullName( FileName : string) : string
11357:  Function WinReboot : boolean
11358:  Function WinDir : String
11359:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11360:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11361:  Function devExecutor : TdevExecutor
11362: end;
11363:
11364: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11365: begin
11366:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11367:  Procedure Associate( Index : integer)
11368:  Procedure UnAssociate( Index : integer)
11369:  Function IsAssociated( Index : integer) : boolean
11370:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11371:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11372:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11373:  procedure RefreshIcons;
```

```
11374: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11375: function MergColor(Colors: Array of TColor): TColor;
11376: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11377: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11378: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11379: function GetInverseColor(AColor: TColor): TColor;
11380: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11381: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11382: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11383: Procedure GetSystemMenuFont(Font: TFont);
11384: end;
11385:
11386: //*************************unit uPSI_JvHLParser;*****************************
11387: function IsStringConstant(const St: string): Boolean;
11388: function IsIntConstant(const St: string): Boolean;
11389: function IsRealConstant(const St: string): Boolean;
11390: function IsIdentifier(const ID: string): Boolean;
11391: function GetStringValue(const St: string): string;
11392: procedure ParseString(const S: string; Ss: TStrings);
11393: function IsStringConstantW(const St: WideString): Boolean;
11394: function IsIntConstantW(const St: WideString): Boolean;
11395: function IsRealConstantW(const St: WideString): Boolean;
11396: function IsIdentifierW(const ID: WideString): Boolean;
11397: function GetStringValueW(const St: WideString): WideString;
11398: procedure ParseStringW(const S: WideString; Ss: TStrings);
11399:
11400:
11401: //*************************unit uPSI_JclMapi;*****************************
11402:
11403: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11404: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11405: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11406: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11407: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11408:
11409: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11410: begin
11411:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11412:  Function BuildType1Message( ADomain, AHost : String) : String
11413:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11414:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11415:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11416: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11417: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11418: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11419: end;
11420:
11421: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11422: begin
11423: ('IpAny','LongWord').SetUInt( $00000000);
11424:  IpLoopBack','LongWord').SetUInt( $7F000001);
11425:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11426:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11427:  PortAny','LongWord( $0000);
11428:  SocketMaxConnections','LongInt'( 5);
11429:  TIpAddr', 'LongWord
11430:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11431:  Function HostToNetLong( HostLong : LongWord) : LongWord
11432:  Function HostToNetShort( HostShort : Word) : Word
11433:  Function NetToHostLong( NetLong : LongWord) : LongWord
11434:  Function NetToHostShort( NetShort : Word) : Word
11435:  Function StrToIp( Ip : string) : TIpAddr
11436:  Function IpToStr( Ip : TIpAddr) : string
11437: end;
11438:
11439: (*----------------------------------------------------------------------------*)
11440: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11441: begin
11442:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11443:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11444:    +'amSha1, AlsmtpClientAuthAutoSelect )
11445:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11446:   SIRegister_TAlSmtpClient(CL);
11447: end;
11448:
11449: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11450: begin
11451: 'TBitNo', 'Integer
11452:   TStByteNo', 'Integer
11453: TStationNo', 'Integer
11454: TInOutNo', 'Integer
11455: TIo', '( EE, AA, NE, NA )
11456: TBitSet', 'set of TBitNo
11457: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11458: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11459: TBitAddr', 'LongInt
```

```
11460: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11461: TByteAddr', 'SmallInt
11462: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11463:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11464:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11465:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
      aBitNo:TBitNo);
11466:  Function BitAddrToStr( Value : TBitAddr) : string
11467:  Function StrToBitAddr( const Value : string) : TBitAddr
11468:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11469:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11470:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11471:  Function ByteAddrToStr( Value : TByteAddr) : string
11472:  Function StrToByteAddr( const Value : string) : TByteAddr
11473:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11474:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11475:  Function InOutStateToStr( State : TInOutState) : string
11476:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11477:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11478: end;
11479:
11480: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11481: begin
11482: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11483:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11484: DpmiPmVector', 'Int64
11485:  'DInterval','LongInt'( 1000);
11486:  //'DEnabled','Boolean')BoolToStr( True);
11487:  'DIntFreq','string' if64
11488:  //'DMessages','Boolean if64);
11489:   SIRegister_TwdxCustomTimer(CL);
11490:   SIRegister_TwdxTimer(CL);
11491:   SIRegister_TwdxRtcTimer(CL);
11492:   SIRegister_TCustomIntTimer(CL);
11493:   SIRegister_TIntTimer(CL);
11494:   SIRegister_TRtcIntTimer(CL);
11495:  Function RealNow : TDateTime
11496:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11497:  Function DateTimeToMs( Time : TDateTime) : LongInt
11498: end;
11499:
11500: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11501: begin
11502: TIdSyslogPRI', 'Integer
11503: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11504:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11505:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11506:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11507:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11508: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11509:   SIRegister_TIdSysLogMsgPart(CL);
11510:   SIRegister_TIdSysLogMessage(CL);
11511:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11512:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11513:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11514:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11515:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11516:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11517: end;
11518:
11519: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11520: begin
11521:  'UWhitespace','String '(?:\s*)
11522:  Function StripSpaces( const AText : string) : string
11523:  Function CharCount( const AText : string; Ch : Char) : Integer
11524:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11525:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11526: end;
11527:
11528:
11529: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11530: begin
11531:  ExtPascalVersion','String '0.9.8
11532:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11533:    +'Opera, brKonqueror, brMobileSafari )
11534:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11535:   AddTypeS('TExtProcedure', 'Procedure
11536:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11537:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11538:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11539:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11540:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11541:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11542:  Function StrToJS( const S : string; UseBR : boolean) : string
11543:  Function CaseOf( const S : string; const Cases : array of string) : integer
11544:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11545:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11546:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11547:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
```

```
11548:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11549:  Function IsUpperCase( S : string) : boolean
11550:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11551:  Function BeautifyCSS( const AStyle : string) : string
11552:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11553:  Function JSDateToDateTime( JSDate : string) : TDateTime
11554: end;
11555:
11556: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11557: begin
11558:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11559:   TSHDeleteOptions', 'set of TSHDeleteOption
11560:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11561:   TSHRenameOptions', 'set of TSHRenameOption
11562:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11563:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11564:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11565:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11566:   TEnumFolderFlags', 'set of TEnumFolderFlag
11567:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11568:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11569:    +'IEnumIdList; Folder : IShellFolder; end
11570:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11571:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11572:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11573:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11574:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11575:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11576:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11577:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11578:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11579:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11580:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11581:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11582:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11583:  Function SHFreeMem( var P : Pointer) : Boolean
11584:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11585:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11586:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11587:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11588:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11589:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11590:  Function PidlFree( var IdList : PItemIdList) : Boolean
11591:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11592:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11593:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11594:  Function PidlToPath( IdList : PItemIdList) : string
11595:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11596:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11597:   PShellLink', '^TShellLink // will not work
11598:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11599:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11600:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11601:  Procedure ShellLinkFree( var Link : TShellLink)
11602:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11603:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11604:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11605:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11606:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11607:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11608:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11609:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11610:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11611:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11612:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11613:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11614:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11615:  Function ShellOpenAs( const FileName : string) : Boolean
11616:  Function ShellRasDial( const EntryName : string) : Boolean
11617:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11618:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11619:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11620:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11621:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11622:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11623:  Function OemKeyScan( wOemChar : Word) : DWORD
11624:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11625: end;
11626:
11627: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11628: begin
11629:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11630:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11631:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11632:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11633:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11634:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11635:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11636:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
```

```
11637:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11638:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11639:  Function xmlValidName( const Text : UnicodeString) : Boolean
11640:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11641:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11642:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11643:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11644:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
      : TUnicodeCodecClass
11645:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11646:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11647:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11648:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11649:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11650:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11651:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11652:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11653:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11654:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11655:  Procedure SelfTestcXMLFunctions
11656: end;
11657:
11658: (*----------------------------------------------------------------------------*)
11659: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11660: begin
11661:  Function AWaitCursor : IUnknown
11662:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11663:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11664:  Function YesNo( const ACaption, AMsg : string) : boolean
11665:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11666:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11667:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11668:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11669:  Procedure GetSystemPaths( Strings : TStrings)
11670:  Procedure MakeEditNumeric( EditHandle : integer)
11671: end;
11672:
11673: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11674: begin
11675:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11676:  'BI_YUY2','LongWord( $32595559);
11677:  'BI_UYVY','LongWord').SetUInt( $59565955);
11678:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11679:  'BI_YVU9','LongWord').SetUInt( $39555659);
11680:  'BI_YUV12','LongWord( $30323449);
11681:  'BI_Y8','LongWord').SetUInt( $20203859);
11682:  'BI_Y211','LongWord').SetUInt( $31313259);
11683:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11684:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11685: end;
11686:
11687: (*----------------------------------------------------------------------------*)
11688: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11689: begin
11690:  'WM_USER','LongWord').SetUInt( $0400);
11691:  'WM_CAP_START','LongWord').SetUint($0400);
11692:  'WM_CAP_END','longword').SetUint($0400+85);
11693:  //WM_CAP_START+  85
11694:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11695:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11696:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11697:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11698:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11699:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11700:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11701:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11702:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11703:  Function capGetUserData( hwnd : THandle) : LongInt
11704:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11705:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11706:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11707:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11708:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11709:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11710:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11711:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11712:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11713:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11714:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11715:  Function capEditCopy( hwnd : THandle) : LongInt
11716:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11717:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11718:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11719:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11720:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11721:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11722:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11723:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11724:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
```

```
11725:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11726:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11727:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11728:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11729:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11730:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11731:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11732:  Function capGrabFrame( hwnd : THandle) : LongInt
11733:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11734:  Function capCaptureSequence( hwnd : THandle) : LongInt
11735:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11736:  Function capCaptureStop( hwnd : THandle) : LongInt
11737:  Function capCaptureAbort( hwnd : THandle) : LongInt
11738:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11739:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11740:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11741:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11742:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11743:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11744:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11745:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11746:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11747:  Function capPalettePaste( hwnd : THandle) : LongInt
11748:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11749:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11750:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11751:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11752:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11753:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11754:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11755:   //PCapStatus', '^TCapStatus // will not work
11756:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11757:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11758:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11759:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11760:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11761:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11762:    +' wNumAudioAllocated : WORD; end
11763:   //PCaptureParms', '^TCaptureParms // will not work
11764:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11765:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11766:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11767:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11768:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11769:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11770:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11771:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11772:    +'he : BOOL; AVStreamMaster : WORD; end
11773:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11774:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11775:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11776:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11777:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
      : Integer; hwndParent : THandle; nID : Integer) : THandle
11778:  Function
      capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11779:  'IDS_CAP_BEGIN','LongInt'( 300);
11780:  'IDS_CAP_END','LongInt'( 301);
11781:  'IDS_CAP_INFO','LongInt'( 401);
11782:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11783:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11784:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11785:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11786:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11787:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11788:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11789:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11790:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11791:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11792:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11793:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11794:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11795:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11796:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11797:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11798:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11799:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11800:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11801:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11802:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11803:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11804:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11805:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11806:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11807:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11808:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11809:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11810:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11811:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
```

```
11812:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11813:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11814:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11815:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11816:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11817:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11818:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11819:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11820:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11821:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11822:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11823:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11824:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11825:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11826:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11827:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11828:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11829:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11830:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11831:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11832:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11833:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11834:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11835:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11836:  'AVICAP32','String 'AVICAP32.dll
11837: end;
11838:
11839: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11840: begin
11841:  Function AlBoolToInt( Value : Boolean) : Integer
11842:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11843:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11844:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11845:  Function ALInc( var x : integer; Count : integer) : Integer
11846:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11847:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11848:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11849:  Function ALIsInteger(const S: AnsiString): Boolean;
11850:  function ALIsDecimal(const S: AnsiString): boolean;
11851:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11852:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11853:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11854:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11855:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11856:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11857:  Function ALRandomStr(const aLength: Longint): AnsiString;
11858:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11859:  Function ALRandomStrU(const aLength: Longint): String;
11860: end;
11861:
11862: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11863: begin
11864:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
       aTrueStr: AnsiString; const aFalseStr : AnsiString)
11865: end;
11866:
11867: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11868: begin
11869:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11870:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11871:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11872:    +'; ullAvailExtendedVirtual : Int64; end
11873:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11874:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11875:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11876:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11877:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11878:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11879:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11880:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11881: end;
11882:
11883: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11884: begin
11885:   SIRegister_THandledObject(CL);
11886:   SIRegister_TEvent(CL);
11887:   SIRegister_TMutex(CL);
11888:   SIRegister_TSharedMem(CL);
11889:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11890:  'TRACE_BUFFER','String 'TRACE_BUFFER
11891:  'TRACE_MUTEX','String 'TRACE_MUTEX
11892:   //PTraceEntry', '^TTraceEntry // will not work
11893:   SIRegister_TIPCTracer(CL);
11894:  'MAX_CLIENTS','LongInt'( 6);
11895:  'IPCTIMEOUT','LongInt'( 2000);
11896:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11897:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11898:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11899:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
```

```
11900:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11901:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11902:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11903:   FindClass('TOBJECT'),'EMonitorActive
11904:   FindClass('TOBJECT'),'TIPCThread
11905:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11906:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11907:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11908:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11909:   TClientFlags', 'set of TClientFlag
11910:   //PEventData', '^TEventData // will not work
11911:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11912:    +'lag; Flags : TClientFlags; end
11913:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11914:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11915:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11916:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11917:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11918:   SIRegister_TIPCEvent(CL);
11919:   //PClientDirRecords', '^TClientDirRecords // will not work
11920:   SIRegister_TClientDirectory(CL);
11921:   TIPCState', '( stInActive, stDisconnected, stConnected )
11922:   SIRegister_TIPCThread(CL);
11923:   SIRegister_TIPCMonitor(CL);
11924:   SIRegister_TIPCClient(CL);
11925:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11926: end;
11927:
11928: (*---------------------------------------------------------------------------*)
11929: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11930: begin
11931:   SIRegister_TAlGSMComm(CL);
11932:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11933:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
      AMessage:AnsiString);
11934:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11935:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
      UseGreekAlphabet:Bool):Widestring;
11936:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11937:  end;
11938:
11939: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11940: begin
11941:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11942:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11943:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11944:   TInternetScheme', 'integer
11945:   TALIPv6Binary', 'array[1..16] of Char;
11946:  // TALIPv6Binary = array[1..16] of ansiChar;
11947:  //   TInternetScheme = Integer;
11948:   SIRegister_TALHTTPRequestHeader(CL);
11949:   SIRegister_TALHTTPCookie(CL);
11950:   SIRegister_TALHTTPCookieCollection(CL);
11951:   SIRegister_TALHTTPResponseHeader(CL);
11952:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11953:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11954: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
      Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11955: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
      Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11956: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
      Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11957:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11958:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11959:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11960:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11961:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11962:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
      ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11963:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
      Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11964:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11965:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11966:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11967:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11968:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11969:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11970:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11971:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11972:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11973:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11974:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
11975:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11976:  Function ALZeroIpV6 : TALIPv6Binary
11977:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11978:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
11979:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
11980:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
11981: end;
```

```
11982:
11983: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11984: begin
11985:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Bool;
11986:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11987:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
11988:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
11989:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
11990:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
11991:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
11992:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
11993:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
11994:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
11995:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
11996: end;
11997:
11998: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11999: begin
12000:   SIRegister_TALEMailHeader(CL);
12001:   SIRegister_TALNewsArticleHeader(CL);
12002:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
12003:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
12004:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
12005:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
12006:  Function AlGenerateInternetMessageID : AnsiString;
12007:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
12008:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
12009:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12010: end;
12011:
12012: (*----------------------------------------------------------------------------*)
12013: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12014: begin
12015:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12016:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
12017:  Function ALgetLocalIPs : TALStrings
12018:  Function ALgetLocalHostName : AnsiString
12019: end;
12020:
12021: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12022: begin
12023:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
12024:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12025:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12026:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
12027:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12028:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12029:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
12030:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12031:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
12032: end;
12033:
12034: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12035: begin
12036:   TStartupInfoA', 'TStartupInfo
12037:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12038:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12039:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12040:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12041:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12042:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12043:  SE_TCB_NAME','String 'SeTcbPrivilege
12044:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12045:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12046:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12047:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12048:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12049:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12050:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12051:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12052:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12053:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12054:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12055:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12056:  SE_DEBUG_NAME','String 'SeDebugPrivilege
```

```
12057:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12058:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12059:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12060:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12061:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12062:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12063:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12064:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12065:  Function AlGetEnvironmentString : AnsiString
12066:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12067:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12068:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12069:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12070:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12071: end;
12072:
12073: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12074: begin
12075:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12076:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12077:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12078:  Function ALGetModuleName : ansistring
12079:  Function ALGetModuleFileNameWithoutExtension : ansistring
12080:  Function ALGetModulePath : ansiString
12081:  Function AlGetFileSize( const AFileName : ansistring) : int64
12082:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12083:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12084:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12085:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12086:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12087:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12088:  Function ALFileExists( const Path : ansiString) : boolean
12089:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12090:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12091:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12092:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12093:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12094: end;
12095:
12096: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12097: begin
12098:   NativeInt', 'Integer
12099:   NativeUInt', 'Cardinal
12100:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12101:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12102:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12103:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12104:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12105:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12106:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12107:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12108:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12109:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12110:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12111:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12112:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12113:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12114:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12115:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12116:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12117:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12118:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
12119:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12120:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12121:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12122:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12123:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12124:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12125:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12126:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12127:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12128:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12129:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
```

```
12130:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12131:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12132: end;
12133:
12134: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12135: begin
12136:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12137:   FindClass('TOBJECT'),'TALXMLNode
12138:   FindClass('TOBJECT'),'TALXMLNodeList
12139:   FindClass('TOBJECT'),'TALXMLDocument
12140:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12141:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12142:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12143:    +'nst Name : AnsiString; const Attributes : TALStrings)
12144:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12145:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12146:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12147:    +'ntDocType, ntDocFragment, ntNotation )
12148:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12149:   TALXMLDocOptions', 'set of TALXMLDocOption
12150:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12151:   TALXMLParseOptions', 'set of TALXMLParseOption
12152:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12153:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12154:   SIRegister_EALXMLDocError(CL);
12155:   SIRegister_TALXMLNodeList(CL);
12156:   SIRegister_TALXMLNode(CL);
12157:   SIRegister_TALXmlElementNode(CL);
12158:   SIRegister_TALXmlAttributeNode(CL);
12159:   SIRegister_TALXmlTextNode(CL);
12160:   SIRegister_TALXmlDocumentNode(CL);
12161:   SIRegister_TALXmlCommentNode(CL);
12162:   SIRegister_TALXmlProcessingInstrNode(CL);
12163:   SIRegister_TALXmlCDataNode(CL);
12164:   SIRegister_TALXmlEntityRefNode(CL);
12165:   SIRegister_TALXmlEntityNode(CL);
12166:   SIRegister_TALXmlDocTypeNode(CL);
12167:   SIRegister_TALXmlDocFragmentNode(CL);
12168:   SIRegister_TALXmlNotationNode(CL);
12169:   SIRegister_TALXMLDocument(CL);
12170:  cAlXMLUTF8EncodingStr','String 'UTF-8
12171:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12172:  CALNSDelim','String ':
12173:  CALXML','String 'xml
12174:  CALVersion','String 'version
12175:  CALEncoding','String 'encoding
12176:  CALStandalone','String 'standalone
12177:  CALDefaultNodeIndent','String '
12178:  CALXmlDocument','String 'DOCUMENT
12179:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12180:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
       EncodingStr:AnsiString);
12181:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
       ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12182:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
       ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12183:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
       Recurse: Boolean):TalxmlNode
12184:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
       AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12185:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
       AnsiString
12186: end;
12187:
12188: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12189: //based on TEEProc, TeCanvas, TEEngine, TChart
12190: begin
12191:  'TeePiStep','Double').setExtended( Pi / 180.0);
12192:  'TeeDefaultPerspective','LongInt'( 100);
12193:  'TeeMinAngle','LongInt'( 270);
12194:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12195:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12196:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12197:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12198:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12199:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12200:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12201:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12202:  'TA_LEFT','LongInt'( 0);
12203:  'TA_RIGHT','LongInt'( 2);
12204:  'TA_CENTER','LongInt'( 6);
12205:  'TA_TOP','LongInt'( 0);
12206:  'TA_BOTTOM','LongInt'( 8);
12207:  'teePATCOPY','LongInt'( 0);
12208:  'NumCirclePoints','LongInt'( 64);
12209:  'teeDEFAULT_CHARSET','LongInt'( 1);
12210:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12211:  'TA_LEFT','LongInt'( 0);
12212:  'bs_Solid','LongInt'( 0);
```

```
12213:  'teepf24Bit','LongInt'( 0);
12214:  'teepfDevice','LongInt'( 1);
12215:  'CM_MOUSELEAVE','LongInt'( 10000);
12216:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12217:  'DC_BRUSH','LongInt'( 18);
12218:  'DC_PEN','LongInt'( 19);
12219:   teeCOLORREF', 'LongWord
12220:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12221:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12222:   SIRegister_TFilterRegion(CL);
12223:   SIRegister_IFormCreator(CL);
12224:   SIRegister_TTeeFilter(CL);
12225:   //TFilterClass', 'class of TTeeFilter
12226:   SIRegister_TFilterItems(CL);
12227:   SIRegister_TConvolveFilter(CL);
12228:   SIRegister_TBlurFilter(CL);
12229:   SIRegister_TTeePicture(CL);
12230:   TPenEndStyle', '( esRound, esSquare, esFlat )
12231:   SIRegister_TChartPen(CL);
12232:   SIRegister_TChartHiddenPen(CL);
12233:   SIRegister_TDottedGrayPen(CL);
12234:   SIRegister_TDarkGrayPen(CL);
12235:   SIRegister_TWhitePen(CL);
12236:   SIRegister_TChartBrush(CL);
12237:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12238:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12239:   SIRegister_TView3DOptions(CL);
12240:   FindClass('TOBJECT'),'TTeeCanvas
12241:   TTeeTransparency', 'Integer
12242:   SIRegister_TTeeBlend(CL);
12243:   FindClass('TOBJECT'),'TCanvas3D
12244:   SIRegister_TTeeShadow(CL);
12245:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
      gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12246:   FindClass('TOBJECT'),'TSubGradient
12247:   SIRegister_TCustomTeeGradient(CL);
12248:   SIRegister_TSubGradient(CL);
12249:   SIRegister_TTeeGradient(CL);
12250:   SIRegister_TTeeFontGradient(CL);
12251:   SIRegister_TTeeFont(CL);
12252:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12253:   TCanvasTextAlign', 'Integer
12254:   TTeeCanvasHandle', 'HDC
12255:   SIRegister_TTeeCanvas(CL);
12256:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12257:   SIRegister_TFloatXYZ(CL);
12258:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
12259:   TRGB', 'record blue: byte; green: byte; red: byte; end
12260:   {TRGB=packed record
12261:     Blue  : Byte;
12262:     Green : Byte;
12263:     Red   : Byte;
12264:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12265:
12266: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12267:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12268: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12269: TCanvas3DPlane', '( cpX, cpY, cpZ )
12270:   //IInterface', 'IUnknown
12271:   SIRegister_TCanvas3D(CL);
12272:   SIRegister_TTeeCanvas3D(CL);
12273:   TTrianglePoints', 'Array[0..2] of TPoint;
12274:   TFourPoints', 'Array[0..3] of TPoint;
12275:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12276:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12277:  Function Point3D( const x, y, z : Integer) : TPoint3D
12278:  Procedure SwapDouble( var a, b : Double)
12279:  Procedure SwapInteger( var a, b : Integer)
12280:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12281:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12282:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12283:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12284:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12285:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12286:  Procedure UnClipCanvas( ACanvas : TCanvas)
12287:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12288:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12289:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12290:  'TeeCharForHeight','String 'W
12291:  'DarkerColorQuantity','Byte').SetUInt( 128);
12292:  'DarkColorQuantity','Byte').SetUInt( 64);
12293:   TButtonGetColorProc', 'Function  : TColor
12294:   SIRegister_TTeeButton(CL);
12295:   SIRegister_TButtonColor(CL);
12296:   SIRegister_TComboFlat(CL);
12297:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12298:  Function TeePoint( const aX, aY : Integer) : TPoint
12299:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12300:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
```

```
12301:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12302:  Function OrientRectangle( const R : TRect) : TRect
12303:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12304:  Function PolygonBounds( const P : array of TPoint) : TRect
12305:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12306:  Function RGBValue( const Color : TColor) : TRGB
12307:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12308:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12309:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12310:  Function TeeCull( const P : TFourPoints) : Boolean;
12311:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12312:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12313:  Procedure SmoothStretch( Src, Dst : TBitmap);
12314:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12315:  Function TeeDistance( const x, y : Double) : Double
12316:  Function TeeLoadLibrary( const FileName : String) : HInst
12317:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12318:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12319:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12320:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
      Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12321:   SIRegister_ICanvasHyperlinks(CL);
12322:   SIRegister_ICanvasToolTips(CL);
12323:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12324: end;
12325:
12326: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12327: begin
12328:   TOvcHdc', 'Integer
12329:   TOvcHWND', 'Cardinal
12330:   TOvcHdc', 'HDC
12331:   TOvcHWND', 'HWND
12332:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12333:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12334:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12335:  Function DefaultEpoch : Integer
12336:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12337:  Procedure FixRealPrim( P : PChar; DC : Char)
12338:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12339:  Function GetLeftButton : Byte
12340:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12341:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12342:  Function GetShiftFlags : Byte
12343:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12344:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12345:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12346:  Function ovIsForegroundTask : Boolean
12347:  Function ovTrimLeft( const S : string) : string
12348:  Function ovTrimRight( const S : string) : string
12349:  Function ovQuotedStr( const S : string) : string
12350:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12351:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12352:  Function PtrDiff( const P1, P2 : PChar) : Word
12353:  Procedure PtrInc( var P, Delta : Word)
12354:  Procedure PtrDec( var P, Delta : Word)
12355:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12356:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
      SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12357:  Function ovMinI( X, Y : Integer) : Integer
12358:  Function ovMaxI( X, Y : Integer) : Integer
12359:  Function ovMinL( X, Y : LongInt) : LongInt
12360:  Function ovMaxL( X, Y : LongInt) : LongInt
12361:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12362:  Function PartialCompare( const S1, S2 : string) : Boolean
12363:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12364:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12365:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12366:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
      TransparentColor : TColor)
12367:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
      TransparentColor : TColorRef)
12368:  Function ovWidthOf( const R : TRect) : Integer
12369:  Function ovHeightOf( const R : TRect) : Integer
12370:  Procedure ovDebugOutput( const S : string)
12371:  Function GetArrowWidth( Width, Height : Integer) : Integer
12372:  Procedure StripCharSeq( CharSeq : string; var Str : string)
12373:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12374:  Procedure StripCharFromFront( aChr : Char; var Str : string)
12375:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12376:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12377:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12378:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12379:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12380:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12381:  Function CreateMetaFile( p1 : PChar) : HDC
12382:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12383:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12384:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12385:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
```

```
12386:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12387:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12388:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12389:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12390:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12391:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12392:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12393:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12394:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
       SrcHeight:Int;Rop:DWORD):BOOL
12395:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12396:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
       SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12397:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12398:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12399:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12400:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12401:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12402:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12403:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12404:  Function UpdateColors( DC : HDC) : BOOL
12405:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12406:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12407:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12408:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12409:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12410:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12411:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
       HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12412:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
       yMask:Int):BOOL;
12413:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12414:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12415:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12416:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12417:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12418:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12419:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12420:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12421:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12422:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12423:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12424:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12425: end;
12426:
12427: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12428: begin
12429:   SIRegister_TOvcAbstractStore(CL);
12430:   //PExPropInfo', '^TExPropInfo // will not work
12431: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12432:   SIRegister_TOvcPropertyList(CL);
12433:   SIRegister_TOvcDataFiler(CL);
12434:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12435:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12436:  Function CreateStoredItem( const CompName, PropName : string) : string
12437:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12438:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12439: end;
12440:
12441: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12442: begin
12443:  'ovsetsize','LongInt'( 16);
12444:  'etSyntax','LongInt'( 0);
12445:  'etSymantic','LongInt'( 1);
12446:  'chCR','Char #13);
12447:  'chLF','Char #10);
12448:  'chLineSeparator',' chCR);
12449:   SIRegister_TCocoError(CL);
12450:   SIRegister_TCommentItem(CL);
12451:   SIRegister_TCommentList(CL);
12452:   SIRegister_TSymbolPosition(CL);
12453: TGenListType', '( glNever, glAlways, glOnError )
12454: TovBitSet', 'set of Integer
12455:   //PStartTable', '^TStartTable // will not work
12456: 'TovCharSet', 'set of AnsiChar
12457: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12458: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12459: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12460: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12461: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12462:    +'osition; const Data : string; ErrorType : integer)
12463: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12464: TGetCH', 'Function ( pos : longint) : char
12465: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12466:   SIRegister_TCocoRScanner(CL);
12467:   SIRegister_TCocoRGrammar(CL);
12468:  '_EF','Char #0);
12469:  '_TAB','Char').SetString( #09);
12470:  '_CR','Char').SetString( #13);
```

```
12471:  '_LF','Char').SetString( #10);
12472:  '_EL','').SetString( _CR);
12473:  '_EOF','Char').SetString( #26);
12474:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12475:  'minErrDist','LongInt'( 2);
12476:  Function ovPadL( S : string; ch : char; L : integer) : string
12477: end;
12478:
12479:   TFormatSettings = record
12480:     CurrencyFormat: Byte;
12481:     NegCurrFormat: Byte;
12482:     ThousandSeparator: Char;
12483:     DecimalSeparator: Char;
12484:     CurrencyDecimals: Byte;
12485:     DateSeparator: Char;
12486:     TimeSeparator: Char;
12487:     ListSeparator: Char;
12488:     CurrencyString: string;
12489:     ShortDateFormat: string;
12490:     LongDateFormat: string;
12491:     TimeAMString: string;
12492:     TimePMString: string;
12493:     ShortTimeFormat: string;
12494:     LongTimeFormat: string;
12495:     ShortMonthNames: array[1..12] of string;
12496:     LongMonthNames: array[1..12] of string;
12497:     ShortDayNames: array[1..7] of string;
12498:     LongDayNames: array[1..7] of string;
12499:     TwoDigitYearCenturyWindow: Word;
12500:   end;
12501:
12502: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12503: begin
12504:  Function ovFormatSettings : TFormatSettings
12505: end;
12506:
12507: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12508: begin
12509:   TOvcCharSet', 'set of Char
12510:   ovBTable', 'array[0..255] of Byte
12511:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
      HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12512:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12513:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12514:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12515:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12516:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12517:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12518:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12519:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12520:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12521:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12522:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12523:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12524:  Function LoCaseChar( C : Char) : Char
12525:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12526:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12527:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12528:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12529:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12530:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12531:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12532:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12533:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12534:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12535:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12536:  Procedure TrimAllSpacesPChar( P : PChar)
12537:  Function TrimEmbeddedZeros( const S : string) : string
12538:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12539:  Function TrimTrailPrimPChar( S : PChar) : PChar
12540:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12541:  Function TrimTrailingZeros( const S : string) : string
12542:  Procedure TrimTrailingZerosPChar( P : PChar)
12543:  Function UpCaseChar( C : Char) : Char
12544:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12545:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12546:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12547: end;
12548:
12549: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12550: begin
12551:   //PRaiseFrame', '^TRaiseFrame // will not work
12552:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12553:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12554:  Procedure SafeCloseHandle( var Handle : THandle)
12555:  Procedure ExchangeInteger( X1, X2 : Integer)
12556:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12557:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12558:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
```

```
12559:
12560: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12561:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12562: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12563:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12564:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12565:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12566:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12567:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12568:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12569:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12570:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12571:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12572:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12573:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12574:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12575:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12576:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12577:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12578:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12579:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12580:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12581:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12582:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12583:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12584:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12585:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12586:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12587:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12588:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12589:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12590:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12591:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12592:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12593:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12594:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12595:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12596:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12597:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12598:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12599:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12600:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12601:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12602:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12603:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12604:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12605:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12606:       var lpLuid: TLargeInteger): BOOL; stdcall;
12607:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12608:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12609:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12610:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12611:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12612:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12613:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12614:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12615:       var GenerateOnClose: BOOL): BOOL; stdcall;
12616:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12617:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12618:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12619:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12620:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12621:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12622:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12623:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12624:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12625:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12626:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12627:       var phkResult: HKEY): Longint; stdcall;
12628:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12629:       var phkResult: HKEY): Longint; stdcall;
12630:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12631:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12632:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12633:       lpdwDisposition: PDWORD): Longint; stdcall;
12634:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12635:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12636:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12637:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12638:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12639:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12640:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12641:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12642:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12643:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12644:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12645:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12646:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12647:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
```

```
12648:       lpcbClass: PDWORD; lpReserved: Pointer;
12649:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12650:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12651:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12652:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12653:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12654:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12655:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12656:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12657:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12658:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12659:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12660:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12661:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12662:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12663:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12664:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12665:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12666:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12667:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12668:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12669:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12670:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12671:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12672:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12673:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12674:
12675:  Function wAddAtom( lpString : PKOLChar) : ATOM
12676:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12677:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
      lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12678:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12679:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
      lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12680:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12681:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12682:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12683:  Function wCreateDirectoryEx(lpTemplateDirectory,
      lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12684:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
      bInitialState:BOOL;lpName:PKOLChar):THandle;
12685:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
      PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12686:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
      dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12687:  Function wCreateHardLink(lpFileName,
      lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12688:  Function
      CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12689:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
      nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12690:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
      lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
      Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
      lpProcessInfo:TProcessInformation):BOOL
12691:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
      Longint; lpName : PKOLChar) : THandle
12692:  Function
      wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12693:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12694:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12695:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12696:  //Function
      wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12697:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12698:  //Function
      wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12699:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12700:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12701:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12702:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12703:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12704:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12705:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
      dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12706:  Function wFindAtom( lpString : PKOLChar) : ATOM
12707:  Function
      wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12708:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12709:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
      Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12710:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12711:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12712:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12713:  Function
      wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12714:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
      DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
```

```
12715:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12716:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12717:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12718:  Function wGetCommandLine : PKOLChar
12719:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12720:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12721:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12722:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
      PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12723:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12724:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
      lpDateStr : PKOLChar; cchDate : Integer) : Integer
12725:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12726:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
      lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12727:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
      lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12728:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12729:  Function wGetEnvironmentStrings : PKOLChar
12730:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12731:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12732:  //Function
      wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12733:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
      lpFilePart:PKOLChar):DWORD;
12734:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12735:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12736:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12737:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12738:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
      lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12739:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
      lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12740:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12741:  Function
      wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12742:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12743:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
      nSize:DWORD; lpFileName : PKOLChar) : DWORD
12744:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12745:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12746:  Function wGetProfileString(lpAppName,lpKeyName,
      lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12747:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12748:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12749: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
      lpCharType):BOOL
12750:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12751:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12752:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12753: //Function
      wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12754: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12755: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
      : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
      lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12756:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12757:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12758:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12759:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12760:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12761:  Function
      wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12762:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12763:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12764:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12765:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12766:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12767:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12768:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12769:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12770:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12771:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12772:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12773:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
      lpNumberOfEventsRead:DWORD):BOOL;
12774:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12775:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12776:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
      lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12777:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
      lpNumbOfEventsRead:DWORD):BOOL;
12778:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
      : TCoord; var lpReadRegion : TSmallRect) : BOOL
12779:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
      DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12780:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12781:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
      lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
```

```
12782:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12783:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12784:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12785:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12786:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12787:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12788:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12789:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12790:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12791:  //Function wUpdateResource(hUpdate:THandle;lpType,
        lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12792:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12793:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12794:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
        DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12795:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
        var lpNumberOfEventsWritten : DWORD) : BOOL
12796:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
        TCoord; var lpWriteRegion : TSmallRect) : BOOL
12797:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12798:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12799:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12800:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12801:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12802:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12803:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12804:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12805:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12806:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12807:  Function wlstrlen( lpString : PKOLChar) : Integer
12808:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
        PNetConnectInfoStruct) : DWORD
12809:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
        lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12810:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12811:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12812:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12813:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12814:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12815:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12816:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12817:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12818:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
        : PKOLChar; nNameBufSize : DWORD) : DWORD
12819:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12820:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12821:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12822: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
        lpBufferSize:DWORD):DWORD;
12823:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12824: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12825: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12826:  //Function wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12827:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12828:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12829:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
        lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12830:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
        szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12831: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12832: //Func wGetPrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12833: //Func wWritePrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12834:  Function wAddFontResource( FileName : PKOLChar) : Integer
12835:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12836:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12837:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12838:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12839:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12840: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12841:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
        fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
        fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12842:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12843:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12844: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12845:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12846:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12847: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
        pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12848: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12849:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12850:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
```

```
12851:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12852:  //Function wExtTextOut(DC:HDC;X,
        Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12853:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12854:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12855:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12856:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12857: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12858: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12859:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12860:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12861: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12862: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
        lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12863:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12864: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12865:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12866: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12867:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12868:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12869:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12870:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12871:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12872:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12873:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12874:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12875:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12876:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12877:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12878:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12879:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12880:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12881:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12882:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12883:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12884:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12885:  //Function
        wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12886:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12887: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
        dwFlags : DWORD; lParam : Pointer) : Longint
12888:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12889:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12890:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12891:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12892:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12893:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12894: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12895:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12896:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12897:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12898:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12899:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12900:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12901:  //Function wCreateDesktop(lpszDesktop,
        lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12902:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
        HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12903:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
        lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12904:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
        hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12905: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
        nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12906:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
        dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12907:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12908:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12909:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12910:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12911:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
        : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12912:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
        : TFNDlgProc; dwInitParam : LPARAM) : Integer
12913:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12914:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
        nIDStaticPath:Integer;uFileType:UINT):Integer;
12915:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
        nIDStaticPath:Int;uFiletype:UINT):Int;
12916:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12917:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12918: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
        cy:Int;Flags:UINT):BOOL;
12919:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12920:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12921:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12922:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
        pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12923: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
```

```
12924:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12925:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12926:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12927:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12928:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12929:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12930:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12931:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12932:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12933:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12934:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12935:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
       lpnTabStopPositions) : DWORD
12936: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
       lpnLengthNeed:DWORD)BOOL;
12937:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12938:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12939:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12940:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12941: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
       nHeigt:Int):BOOL;
12942:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12943:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12944:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12945:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12946:  Function wIsCharLower( ch : KOLChar) : BOOL
12947:  Function wIsCharUpper( ch : KOLChar) : BOOL
12948:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12949:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12950:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12951:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12952:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12953:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12954:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12955:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12956:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
12957:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12958:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12959:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
12960:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
12961:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
12962:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12963:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
12964:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12965:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
12966:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
12967:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12968:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12969:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12970:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
12971:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
12972:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12973:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12974:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
12975: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
12976: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
12977:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
12978: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12979:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
12980:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
12981:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12982:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12983:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
       lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
12984:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
       lpdwResult:DWORD): LRESULT
12985:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12986:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
12987:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
12988:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12989:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
12990: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
12991:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
12992:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
12993:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
12994:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
12995: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
12996:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
       lpnTabStopPositions,nTabOrigin:Int):Longint;
12997:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
12998:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
12999:  Function wVkKeyScan( ch : KOLChar) : SHORT
13000:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
13001:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
13002:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
13003:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
13004:
13005: //TestDrive!
13006: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
```

```
13007:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
13008:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
13009:  Function GetLocalUserSidStr( const UserName : string) : string
13010:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
13011:  Function Impersonate2User( const domain : string; const user : string) : boolean
13012:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
13013:  Function KillProcessbyname( const exename : string; var found : integer) : integer
13014:  Function getWinProcessList : TStringList
13015:  function WaitTilClose(hWnd: Integer): Integer;
13016:  function DoUserMsgs: Boolean;
13017:  function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13018:  procedure ShowMsg(hParent: Integer; const Mess, Title: String);  //modal but NOT blockable
13019:  procedure DeleteMsgForm(Handle: Integer);
13020:  procedure DisableForms;
13021:  function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13022: end;
13023:
13024: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13025: begin
13026:  'AfMaxSyncSlots','LongInt'( 64);
13027:  'AfSynchronizeTimeout','LongInt'( 2000);
13028:   TAfSyncSlotID', 'DWORD
13029:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13030:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
13031:   TAfSafeDirectSyncEvent', 'Procedure
13032:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13033:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
13034:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
13035:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
13036:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
13037:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
13038:  Function AfIsSyncMethod : Boolean
13039:  Function AfSyncWnd : HWnd
13040:  Function AfSyncStatistics : TAfSyncStatistics
13041:  Procedure AfClearSyncStatistics
13042: end;
13043:
13044: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13045: begin
13046:  'fBinary','LongWord')( $00000001);
13047:  'fParity','LongWord'( $00000002);
13048:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13049:  'fOutxDsrFlow','LongWord')( $00000008);
13050:  'fDtrControl','LongWord')( $00000030);
13051:  'fDtrControlDisable','LongWord')( $00000000);
13052:  'fDtrControlEnable','LongWord')( $00000010);
13053:  'fDtrControlHandshake','LongWord')( $00000020);
13054:  'fDsrSensitivity','LongWord')( $00000040);
13055:  'fTXContinueOnXoff','LongWord')( $00000080);
13056:  'fOutX','LongWord')( $00000100);
13057:  'fInX','LongWord')( $00000200);
13058:  'fErrorChar','LongWord')( $00000400);
13059:  'fNull','LongWord')( $00000800);
13060:  'fRtsControl','LongWord')( $00003000);
13061:  'fRtsControlDisable','LongWord')( $00000000);
13062:  'fRtsControlEnable','LongWord')( $00001000);
13063:  'fRtsControlHandshake','LongWord')( $00002000);
13064:  'fRtsControlToggle','LongWord')( $00003000);
13065:  'fAbortOnError','LongWord')( $00004000);
13066:  'fDummy2','LongWord')( $FFFF8000);
13067:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13068:   FindClass('TOBJECT'),'EAfComPortCoreError
13069:   FindClass('TOBJECT'),'TAfComPortCore
13070:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
13071:    +'tKind : TAfCoreEvent; Data : DWORD)
13072:   SIRegister_TAfComPortCoreThread(CL);
13073:   SIRegister_TAfComPortEventThread(CL);
13074:   SIRegister_TAfComPortWriteThread(CL);
13075:   SIRegister_TAfComPortCore(CL);
13076:  Function FormatDeviceName( PortNumber : Integer) : string
13077: end;
13078:
13079: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13080: begin
13081:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13082:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13083:   SIRegister_TApplicationFileIO(CL);
13084:   TDataFileCapability', '( dfcRead, dfcWrite )
13085:   TDataFileCapabilities', 'set of TDataFileCapability
13086:   SIRegister_TDataFile(CL);
13087:   //TDataFileClass', 'class of TDataFile
13088:  Function ApplicationFileIODefined : Boolean
13089:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13090:  Function FileStreamExists(const fileName: String) : Boolean
13091:  //Procedure Register
13092: end;
13093:
13094: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13095: begin
```

```
13096:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13097:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13098:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13099:   TALFBXScale', 'Integer
13100:   FindClass('TOBJECT'),'EALFBXConvertError
13101:   SIRegister_EALFBXError(CL);
13102:   SIRegister_EALFBXException(CL);
13103:   FindClass('TOBJECT'),'EALFBXGFixError
13104:   FindClass('TOBJECT'),'EALFBXDSQLError
13105:   FindClass('TOBJECT'),'EALFBXDynError
13106:   FindClass('TOBJECT'),'EALFBXGBakError
13107:   FindClass('TOBJECT'),'EALFBXGSecError
13108:   FindClass('TOBJECT'),'EALFBXLicenseError
13109:   FindClass('TOBJECT'),'EALFBXGStatError
13110:   //EALFBXExceptionClass', 'class of EALFBXError
13111:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13112:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13113:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13114:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13115:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13116:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13117:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13118:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13119:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13120:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13121:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13122:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13123:   TALFBXTransParams', 'set of TALFBXTransParam
13124:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13125:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13126:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13127: 'cALFBXMaxParamLength','LongInt'( 125);
13128:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13129:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13130:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13131:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13132:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13133:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13134:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13135:   SIRegister_TALFBXSQLDA(CL);
13136:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13137:   SIRegister_TALFBXPoolStream(CL);
13138:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13139:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13140:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13141:   //TALFBXArrayDesc', 'TISCArrayDesc
13142:   //TALFBXBlobDesc', 'TISCBlobDesc
13143:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13144:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13145:   SIRegister_TALFBXSQLResult(CL);
13146:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13147:   SIRegister_TALFBXSQLParams(CL);
13148:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13149:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13150:    +'atementType : TALFBXStatementType; end
13151:   FindClass('TOBJECT'),'TALFBXLibrary
13152:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13153:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13154:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13155:    //+' Excep : EALFBXExceptionClass)
13156:   SIRegister_TALFBXLibrary(CL);
13157:  'cAlFBXDateOffset','LongInt'( 15018);
13158:  'cALFBXTimeCoeff','LongInt'( 864000000);
13159:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13160:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13161:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13162:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13163:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13164:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13165:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13166:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13167:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13168:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13169:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13170:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13171:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13172:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13173: end;
13174:
13175: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13176: begin
13177:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13178:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13179:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13180:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13181:    +'teger; First : Integer; CacheThreshold : Integer; end
13182:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13183:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13184:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
```

```
13185:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13186:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13187:   SIRegister_TALFBXClient(CL);
13188:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13189:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13190:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13191:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13192:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13193:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13194:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13195:   SIRegister_TALFBXConnectionPoolClient(CL);
13196:   SIRegister_TALFBXEventThread(CL);
13197:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13198: end;
13199:
13200: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13201: begin
13202: _OSVERSIONINFOA = record
13203:     dwOSVersionInfoSize: DWORD;
13204:     dwMajorVersion: DWORD;
13205:     dwMinorVersion: DWORD;
13206:     dwBuildNumber: DWORD;
13207:     dwPlatformId: DWORD;
13208:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13209:   end;
13210:  TOSVersionInfoA', '_OSVERSIONINFOA
13211:   TOSVersionInfo', 'TOSVersionInfoA
13212:  'WS_EX_RIGHT','LongWord')( $00001000);
13213:  'WS_EX_LEFT','LongWord')( $00000000);
13214:  'WS_EX_RTLREADING','LongWord')( $00002000);
13215:  'WS_EX_LTRREADING','LongWord')( $00000000);
13216:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13217:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13218:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13219:  'LAYOUT_RTL','LongWord')( $00000001);
13220:  'LAYOUT_BTT','LongWord')( $00000002);
13221:  'LAYOUT_VBH','LongWord')( $00000004);
13222:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13223:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13224:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13225:  Function GetLayout( dc : hdc) : DWORD
13226:  Function IsBidi : Boolean
13227:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13228:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13229:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13230:  Function GetPriorityClass( hProcess : THandle) : DWORD
13231:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13232:  Function CloseClipboard : BOOL
13233:  Function GetClipboardSequenceNumber : DWORD
13234:  Function GetClipboardOwner : HWND
13235:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13236:  Function GetClipboardViewer : HWND
13237:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13238:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13239:  Function GetClipboardData( uFormat : UINT) : THandle
13240:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13241:  Function CountClipboardFormats : Integer
13242:  Function EnumClipboardFormats( format : UINT) : UINT
13243:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13244:  Function EmptyClipboard : BOOL
13245:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13246:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13247:  Function GetOpenClipboardWindow : HWND
13248:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13249:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13250:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13251:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13252:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13253:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13254:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13255:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13256:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13257: end;
13258:
13259: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13260: begin
13261:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13262:  Function GetTemporaryFilesPath : String
13263:  Function GetTemporaryFileName : String
13264:  Function FindFileInPaths( const fileName, paths : String) : String
13265:  Function PathsToString( const paths : TStrings) : String
13266:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13267:  //Function MacroExpandPath( const aPath : String) : String
13268: end;
13269:
13270: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13271: begin
13272:   SIRegister_TALMultiPartBaseContent(CL);
13273:   SIRegister_TALMultiPartBaseContents(CL);
```

```
13274:   SIRegister_TAlMultiPartBaseStream(CL);
13275:   SIRegister_TALMultipartBaseEncoder(CL);
13276:   SIRegister_TALMultipartBaseDecoder(CL);
13277:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13278:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13279:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13280: end;
13281:
13282: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13283: begin
13284:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13285:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13286:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13287:  Function aAllocPadedMem( Size : Cardinal) : TObject
13288:  Procedure aFreePadedMem( var P : TObject);
13289:  Procedure aFreePadedMem1( var P : PChar);
13290:  Function aCheckPadedMem( P : Pointer) : Byte
13291:  Function aGetPadMemSize( P : Pointer) : Cardinal
13292:  Function aAllocMem( Size : Cardinal) : Pointer
13293:  Function aStrLen( const Str : PChar) : Cardinal
13294:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13295:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13296:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13297:  Function aStrEnd( const Str : PChar) : PChar
13298:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13299:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13300:  Function aPCharLength( const Str : PChar) : Cardinal
13301:  Function aPCharUpper( Str : PChar) : PChar
13302:  Function aPCharLower( Str : PChar) : PChar
13303:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13304:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13305:  Function aCopyTail( const S : String; Len : Integer) : String
13306:  Function aInt2Thos( I : Int64) : String
13307:  Function aUpperCase( const S : String) : String
13308:  Function aLowerCase( const S : string) : String
13309:  Function aCompareText( const S1, S2 : string) : Integer
13310:  Function aSameText( const S1, S2 : string) : Boolean
13311:  Function aInt2Str( Value : Int64) : String
13312:  Function aStr2Int( const Value : String) : Int64
13313:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13314:  Function aGetFileExt( const FileName : String) : String
13315:  Function aGetFilePath( const FileName : String) : String
13316:  Function aGetFileName( const FileName : String) : String
13317:  Function aChangeExt( const FileName, Extension : String) : String
13318:  Function aAdjustLineBreaks( const S : string) : string
13319:  Function aGetWindowStr( WinHandle : HWND) : String
13320:  Function aDiskSpace( Drive : String) : TdriveSize
13321:  Function aFileExists( FileName : String) : Boolean
13322:  Function aFileSize( FileName : String) : Int64
13323:  Function aDirectoryExists( const Name : string) : Boolean
13324:  Function aSysErrorMessage( ErrorCode : Integer) : string
13325:  Function aShortPathName( const LongName : string) : string
13326:  Function aGetWindowVer : TWinVerRec
13327:  procedure InitDriveSpacePtr;
13328: end;
13329:
13330: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13331: begin
13332:  aZero','LongInt'( 0);
13333:  'makeappDEF','LongInt'( - 1);
13334:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13335:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13336:  'CS_KEYCVTWINDOW','LongInt'( 4);
13337:  'CS_DBLCLKS','LongInt'( 8);
13338:  'CS_OWNDC','LongWord')( $20);
13339:  'CS_CLASSDC','LongWord')( $40);
13340:  'CS_PARENTDC','LongWord')( $80);
13341:  'CS_NOKEYCVT','LongWord')( $100);
13342:  'CS_NOCLOSE','LongWord')( $200);
13343:  'CS_SAVEBITS','LongWord')( $800);
13344:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13345:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13346:  'CS_GLOBALCLASS','LongWord')( $4000);
13347:  'CS_IME','LongWord')( $10000);
13348:  'CS_DROPSHADOW','LongWord')( $20000);
13349:   //PPanelFunc', '^TPanelFunc // will not work
13350:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13351:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13352:   TFontLooks', 'set of TFontLook
13353:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13354:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13355:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13356:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13357:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13358:  Procedure RunMsgLoop( Show : Boolean)
13359:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13360:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
       ID_Number:Cardinal;hFont:Int):Int;
13361:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
```

```
13362:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13363:  Function MakePanel(Left,Top,Width,Height,
        hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13364:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13365:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13366:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13367: end;
13368:
13369: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13370: begin
13371:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13372:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13373:   TScreenSaverOptions', 'set of TScreenSaverOption
13374:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
        ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13375:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13376:   SIRegister_TScreenSaver(CL);
13377:  //Procedure Register
13378:  Procedure SetScreenSaverPassword
13379: end;
13380:
13381: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13382: begin
13383:   FindClass('TOBJECT'),'TXCollection
13384:   SIRegister_EFilerException(CL);
13385:   SIRegister_TXCollectionItem(CL);
13386:   //TXCollectionItemClass', 'class of TXCollectionItem
13387:   SIRegister_TXCollection(CL);
13388:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13389:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13390:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13391:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13392:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13393:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13394: end;
13395:
13396: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13397: begin
13398:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13399:  Procedure xglMapTexCoordToNull
13400:  Procedure xglMapTexCoordToMain
13401:  Procedure xglMapTexCoordToSecond
13402:  Procedure xglMapTexCoordToDual
13403:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13404:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13405:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13406:  Procedure xglBeginUpdate
13407:  Procedure xglEndUpdate
13408:  Procedure xglPushState
13409:  Procedure xglPopState
13410:  Procedure xglForbidSecondTextureUnit
13411:  Procedure xglAllowSecondTextureUnit
13412:  Function xglGetBitWiseMapping : Cardinal
13413: end;
13414:
13415: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13416: begin
13417:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13418:   TBaseListOptions', 'set of TBaseListOption
13419:   SIRegister_TBaseList(CL);
13420:   SIRegister_TBaseVectorList(CL);
13421:   SIRegister_TAffineVectorList(CL);
13422:   SIRegister_TVectorList(CL);
13423:   SIRegister_TTexPointList(CL);
13424:   SIRegister_TXIntegerList(CL);
13425:   //PSingleArrayList', '^TSingleArrayList // will not work
13426:   SIRegister_TSingleList(CL);
13427:   SIRegister_TByteList(CL);
13428:   SIRegister_TQuaternionList(CL);
13429:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13430:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13431:  Procedure FastQuickSortLists(startIndex,
        endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13432: end;
13433:
13434: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13435: begin
13436:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13437:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13438:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
        indices:TIntegerList;list:TAffineVectorList);
13439:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
        indices:TIntegerList;list:TAffineVectorList);
13440:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
        normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13441:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13442:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13443:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13444:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
```

```
13445:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13446:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13447:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13448:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13449:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13450:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
       edgesTriangles : TIntegerList) : TIntegerList
13451:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13452:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
       TPersistentObjectList;
13453:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13454:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
       TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13455: end;
13456:
13457: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13458: begin
13459:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13460:  Procedure FreeMemAndNil( var P : TObject)
13461:  Function PCharOrNil( const S : string) : PChar
13462:   SIRegister_TJclReferenceMemoryStream(CL);
13463:   FindClass('TOBJECT'),'EJclVMTError
13464:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13465:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13466:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13467:   PDynamicIndexList', '^TDynamicIndexList // will not work
13468:   PDynamicAddressList', '^TDynamicAddressList // will not work
13469:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13470:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13471:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13472:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13473:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13474:  Function GetInitTable( AClass : TClass) : PTypeInfo
13475:   PFieldEntry', '^TFieldEntry // will not work}
13476:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13477:  Function JIsClass( Address : Pointer) : Boolean
13478:  Function JIsObject( Address : Pointer) : Boolean
13479:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13480:   TDigitCount', 'Integer
13481:   SIRegister_TJclNumericFormat(CL);
13482:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13483:   TTextHandler', 'Procedure ( const Text : string)
13484: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13485:  Function JExecute(const
       CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13486:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13487:  Function ReadKey : Char   //to and from the DOS console !
13488:   TModuleHandle', 'HINST
13489:   //TModuleHandle', 'Pointer
13490:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13491:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13492:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13493:  Procedure UnloadModule( var Module : TModuleHandle)
13494:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13495:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13496:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13497:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13498:   FindClass('TOBJECT'),'EJclConversionError
13499:  Function JStrToBoolean( const S : string) : Boolean
13500:  Function JBooleanToStr( B : Boolean) : string
13501:  Function JIntToBool( I : Integer) : Boolean
13502:  Function JBoolToInt( B : Boolean) : Integer
13503:  'ListSeparator','String ';
13504:  'ListSeparator1','String ':
13505:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13506:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13507:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13508:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13509:  Function ListItemCount( const List, Separator : string) : Integer
13510:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13511:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13512:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13513:  Function SystemTObjectInstance : LongWord
13514:  Function IsCompiledWithPackages : Boolean
13515:  Function JJclGUIDToString( const GUID : TGUID) : string
13516:  Function JJclStringToGUID( const S : string) : TGUID
13517:   SIRegister_TJclIntfCriticalSection(CL);
13518:   SIRegister_TJclSimpleLog(CL);
13519:  Procedure InitSimpleLog( const ALogFileName : string)
13520: end;
13521:
13522: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13523: begin
13524:   FindClass('TOBJECT'),'EJclBorRADException
13525:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13526:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13527:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13528:   TJclBorRADToolPath', 'string
13529:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
```

```
13530:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13531:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13532:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13533:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13534:  BorRADToolRepositoryFormsPage','String 'Forms
13535:  BorRADToolRepositoryProjectsPage','String 'Projects
13536:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13537:  BorRADToolRepositoryObjectType','String 'Type
13538:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13539:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13540:  BorRADToolRepositoryObjectName','String 'Name
13541:  BorRADToolRepositoryObjectPage','String 'Page
13542:  BorRADToolRepositoryObjectIcon','String 'Icon
13543:  BorRADToolRepositoryObjectDescr','String 'Description
13544:  BorRADToolRepositoryObjectAuthor','String 'Author
13545:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13546:  BorRADToolRepositoryObjectDesigner','String 'Designer
13547:  BorRADToolRepositoryDesignerDfm','String 'dfm
13548:  BorRADToolRepositoryDesignerXfm','String 'xfm
13549:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13550:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13551:  SourceExtensionDelphiPackage','String '.dpk
13552:  SourceExtensionBCBPackage','String '.bpk
13553:  SourceExtensionDelphiProject','String '.dpr
13554:  SourceExtensionBCBProject','String '.bpr
13555:  SourceExtensionBDSProject','String '.bdsproj
13556:  SourceExtensionDProject','String '.dproj
13557:  BinaryExtensionPackage','String '.bpl
13558:  BinaryExtensionLibrary','String '.dll
13559:  BinaryExtensionExecutable','String '.exe
13560:  CompilerExtensionDCP','String '.dcp
13561:  CompilerExtensionBPI','String '.bpi
13562:  CompilerExtensionLIB','String '.lib
13563:  CompilerExtensionTDS','String '.tds
13564:  CompilerExtensionMAP','String '.map
13565:  CompilerExtensionDRC','String '.drc
13566:  CompilerExtensionDEF','String '.def
13567:  SourceExtensionCPP','String '.cpp
13568:  SourceExtensionH','String '.h
13569:  SourceExtensionPAS','String '.pas
13570:  SourceExtensionDFM','String '.dfm
13571:  SourceExtensionXFM','String '.xfm
13572:  SourceDescriptionPAS','String 'Pascal source file
13573:  SourceDescriptionCPP','String 'C++ source file
13574:  DesignerVCL','String 'VCL
13575:  DesignerCLX','String 'CLX
13576:  ProjectTypePackage','String 'package
13577:  ProjectTypeLibrary','String 'library
13578:  ProjectTypeProgram','String 'program
13579:  Personality32Bit','String '32 bit
13580:  Personality64Bit','String '64 bit
13581:  PersonalityDelphi','String 'Delphi
13582:  PersonalityDelphiDotNet','String 'Delphi.net
13583:  PersonalityBCB','String 'C++Builder
13584:  PersonalityCSB','String 'C#Builder
13585:  PersonalityVB','String 'Visual Basic
13586:  PersonalityDesign','String 'Design
13587:  PersonalityUnknown','String 'Unknown personality
13588:  PersonalityBDS','String 'Borland Developer Studio
13589:  DOFDirectoriesSection','String 'Directories
13590:  DOFUnitOutputDirKey','String 'UnitOutputDir
13591:  DOFSearchPathName','String 'SearchPath
13592:  DOFConditionals','String 'Conditionals
13593:  DOFLinkerSection','String 'Linker
13594:  DOFPackagesKey','String 'Packages
13595:  DOFCompilerSection','String 'Compiler
13596:  DOFPackageNoLinkKey','String 'PackageNoLink
13597:  DOFAdditionalSection','String 'Additional
13598:  DOFOptionsKey','String 'Options
13599:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13600:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13601:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13602:   TJclBorPersonalities', 'set of TJclBorPersonality
13603:   TJclBorDesigner', '( bdVCL, bdCLX )
13604:   TJclBorDesigners', 'set of TJClBorDesigner
13605:   TJclBorPlatform', '( bp32bit, bp64bit )
13606:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13607:   SIRegister_TJclBorRADToolInstallationObject(CL);
13608:   SIRegister_TJclBorlandOpenHelp(CL);
13609:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13610:   TJclHelp2Objects', 'set of TJclHelp2Object
13611:   SIRegister_TJclHelp2Manager(CL);
13612:   SIRegister_TJclBorRADToolIdeTool(CL);
13613:   SIRegister_TJclBorRADToolIdePackages(CL);
13614:   SIRegister_IJclCommandLineTool(CL);
13615:   FindClass('TOBJECT'),'EJclCommandLineToolError
13616:   SIRegister_TJclCommandLineTool(CL);
13617:   SIRegister_TJclBorlandCommandLineTool(CL);
13618:   SIRegister_TJclBCC32(CL);
```

```
13619:   SIRegister_TJclDCC32(CL);
13620:   TJclDCC', 'TJclDCC32
13621:   SIRegister_TJclBpr2Mak(CL);
13622:   SIRegister_TJclBorlandMake(CL);
13623:   SIRegister_TJclBorRADToolPalette(CL);
13624:   SIRegister_TJclBorRADToolRepository(CL);
13625:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13626:   TCommandLineTools', 'set of TCommandLineTool
13627:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13628:   SIRegister_TJclBorRADToolInstallation(CL);
13629:   SIRegister_TJclBCBInstallation(CL);
13630:   SIRegister_TJclDelphiInstallation(CL);
13631:   SIRegister_TJclDCCIL(CL);
13632:   SIRegister_TJclBDSInstallation(CL);
13633:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13634:   SIRegister_TJclBorRADToolInstallations(CL);
13635:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13636:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13637:  Function IsDelphiPackage( const FileName : string) : Boolean
13638:  Function IsDelphiProject( const FileName : string) : Boolean
13639:  Function IsBCBPackage( const FileName : string) : Boolean
13640:  Function IsBCBProject( const FileName : string) : Boolean
13641:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13642:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13643:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
       Descript:PString;
13644:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString
13645:  function SamePath(const Path1, Path2: string): Boolean;
13646: end;
13647:
13648: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13649: begin
13650:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13651:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13652:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13653:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13654:  'LPathSeparator','String '/
13655:  'LDirDelimiter','String '/
13656:  'LDirSeparator','String ':
13657:  'JXPathDevicePrefix','String '\\.\
13658:  'JXPathSeparator','String '\
13659:  'JXDirDelimiter','String '\
13660:  'JXDirSeparator','String ';
13661:  'JXPathUncPrefix','String '\\
13662:  'faNormalFile','LongWord')( $00000080);
13663:  //'faUnixSpecific',' faSymLink);
13664:   JXTCompactPath', '( cpCenter, cpEnd )
13665:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13666:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13667:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13668:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13669:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13670:
13671:  Function jxPathAddSeparator( const Path : string) : string
13672:  Function jxPathAddExtension( const Path, Extension : string) : string
13673:  Function jxPathAppend( const Path, Append : string) : string
13674:  Function jxPathBuildRoot( const Drive : Byte) : string
13675:  Function jxPathCanonicalize( const Path : string) : string
13676:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13677:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13678:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13679:  Function jxPathExtractFileDirFixed( const S : string) : string
13680:  Function jxPathExtractFileNameNoExt( const Path : string) : string
13681:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13682:  Function jxPathGetDepth( const Path : string) : Integer
13683:  Function jxPathGetLongName( const Path : string) : string
13684:  Function jxPathGetShortName( const Path : string) : string
13685:  Function jxPathGetLongName( const Path : string) : string
13686:  Function jxPathGetShortName( const Path : string) : string
13687:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13688:  Function jxPathGetTempPath : string
13689:  Function jxPathIsAbsolute( const Path : string) : Boolean
13690:  Function jxPathIsChild( const Path, Base : string) : Boolean
13691:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13692:  Function jxPathIsUNC( const Path : string) : Boolean
13693:  Function jxPathRemoveSeparator( const Path : string) : string
13694:  Function jxPathRemoveExtension( const Path : string) : string
13695:  Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13696:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13697:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13698:   JxTFileListOptions', 'set of TFileListOption
13699:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13700:   TFileHandler', 'Procedure ( const FileName : string)
13701:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13702:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13703:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
       AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
       FileMatchFunc:TFileMatchFunc):Bool;
```

```
13704:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13705:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13706:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13707:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13708:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
       RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13709:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
       IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13710:  Procedure jxCreateEmptyFile( const FileName : string)
13711:  Function jxCloseVolume( var Volume : THandle) : Boolean
13712:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13713:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13714:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13715:  Function jxDelTree( const Path : string) : Boolean
13716:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13717:  Function jxDiskInDrive( Drive : Char) : Boolean
13718:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13719:  Function jxFileCreateTemp( var Prefix : string) : THandle
13720:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13721:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13722:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13723:  Function jxFileExists( const FileName : string) : Boolean
13724:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13725:  Function jxFileRestore( const FileName : string) : Boolean
13726:  Function jxGetBackupFileName( const FileName : string) : string
13727:  Function jxIsBackupFileName( const FileName : string) : Boolean
13728:  Function jxFileGetDisplayName( const FileName : string) : string
13729:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13730:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13731:  Function jxFileGetSize( const FileName : string) : Int64
13732:  Function jxFileGetTempName( const Prefix : string) : string
13733:  Function jxFileGetTypeName( const FileName : string) : string
13734:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13735:  Function jxForceDirectories( Name : string) : Boolean
13736:  Function jxGetDirectorySize( const Path : string) : Int64
13737:  Function jxGetDriveTypeStr( const Drive : Char) : string
13738:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13739:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13740:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13741:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13742:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13743:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
       ResolveSymLinks:Boolean):Integer
13744:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13745:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13746:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13747:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13748:  Function jxGetFileCreation( const FName : string) : TFileTime;
13749:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13750:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13751:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13752:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13753:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13754:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13755:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13756:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13757:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13758:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13759:  Function jxGetModulePath( const Module : HMODULE) : string
13760:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13761:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13762:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13763:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13764:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13765:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13766:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13767:  Function jxOpenVolume( const Drive : Char) : THandle
13768:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13769:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13770:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13771:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13772:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13773:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13774:  Procedure jxShredFile( const FileName : string; Times : Integer)
13775:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13776:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13777:  Function jxSymbolicLinkTarget( const Name : string) : string
13778:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13779:   SIRegister_TJclCustomFileAttrMask(CL);
13780:   SIRegister_TJclFileAttributeMask(CL);
13781:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13782:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13783:   TFileSearchOptions', 'set of TFileSearchOption
13784:   TFileSearchTaskID', 'Integer
13785:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13786:    +'hTaskID; const Aborted : Boolean)
13787:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13788:   SIRegister_IJclFileEnumerator(CL);
13789:   SIRegister_TJclFileEnumerator(CL);
```

```
13790:  Function JxFileSearch : IJclFileEnumerator
13791:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13792:   JxTFileFlags', 'set of TFileFlag
13793:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13794:   SIRegister_TJclFileVersionInfo(CL);
13795:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13796:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13797:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13798:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13799:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13800:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13801:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13802:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13803:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13804:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13805:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13806:   SIRegister_TJclTempFileStream(CL);
13807:   FindClass('TOBJECT'),'TJclCustomFileMapping
13808:   SIRegister_TJclFileMappingView(CL);
13809:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13810:   SIRegister_TJclCustomFileMapping(CL);
13811:   SIRegister_TJclFileMapping(CL);
13812:   SIRegister_TJclSwapFileMapping(CL);
13813:   SIRegister_TJclFileMappingStream(CL);
13814:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13815:   //PPCharArray', '^TPCharArray // will not work
13816:   SIRegister_TJclMappedTextReader(CL);
13817:   SIRegister_TJclFileMaskComparator(CL);
13818:   FindClass('TOBJECT'),'EJclPathError
13819:   FindClass('TOBJECT'),'EJclFileUtilsError
13820:   FindClass('TOBJECT'),'EJclTempFileStreamError
13821:   FindClass('TOBJECT'),'EJclTempFileStreamError
13822:   FindClass('TOBJECT'),'EJclFileMappingError
13823:   FindClass('TOBJECT'),'EJclFileMappingViewError
13824:  Function jxPathGetLongName2( const Path : string) : string
13825:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13826:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13827:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13828:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13829:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13830:  Procedure jxPathListAddItems( var List : string; const Items : string)
13831:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13832:  Procedure jxPathListDelItems( var List : string; const Items : string)
13833:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13834:  Function jxPathListItemCount( const List : string) : Integer
13835:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13836:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13837:  Function jxPathListItemIndex( const List, Item : string) : Integer
13838:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13839:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13840:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13841:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string) : Integer
13842: end;
13843:
13844: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13845: begin
13846:  'UTF8FileHeader','String #$ef#$bb#$bf);
13847:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13848:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13849:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13850:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13851:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13852:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13853:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13854:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13855:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13856:  Function lFileIsReadable( const AFilename : string) : boolean
13857:  Function lFileIsWritable( const AFilename : string) : boolean
13858:  Function lFileIsText( const AFilename : string) : boolean
13859:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13860:  Function lFileIsExecutable( const AFilename : string) : boolean
13861:  Function lFileIsSymlink( const AFilename : string) : boolean
13862:  Function lFileIsHardLink( const AFilename : string) : boolean
13863:  Function lFileSize( const Filename : string) : int64;
13864:  Function lGetFileDescription( const AFilename : string) : string
13865:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13866:  Function lTryReadAllLinks( const Filename : string) : string
13867:  Function lDirPathExists( const FileName : String) : Boolean
13868:  Function lForceDirectory( DirectoryName : string) : boolean
13869:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13870:  Function lProgramDirectory : string
13871:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13872:  Function lExtractFileNameOnly( const AFilename : string) : string
13873:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13874:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
```

```
13875:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13876:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13877:  Function lAppendPathDelim( const Path : string) : string
13878:  Function lChompPathDelim( const Path : string) : string
13879:  Function lTrimFilename( const AFilename : string) : string
13880:  Function lCleanAndExpandFilename( const Filename : string) : string
13881:  Function lCleanAndExpandDirectory( const Filename : string) : string
13882:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13883:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
       AlwaysRequireSharedBaseFolder : Boolean) : string
13884:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13885:  Function lFileIsInPath( const Filename, Path : string) : boolean
13886:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13887:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13888:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13889:  'AllDirectoryEntriesMask','String '*
13890:  Function lGetAllFilesMask : string
13891:  Function lGetExeExt : string
13892:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
       TSearchFileInPathFlags) : string
13893:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
       TSearchFileInPathFlags) : TStrings
13894:  Function lFindDiskFilename( const Filename : string) : string
13895:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13896:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13897:  Function lGetDarwinSystemFilename( Filename : string) : string
13898:   SIRegister_TFileIterator(CL);
13899:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13900:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13901:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13902:   SIRegister_TFileSearcher(CL);
13903:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13904:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13905:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13906:  // TCopyFileFlags', 'set of TCopyFileFlag
13907:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13908:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13909:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13910:  Function lReadFileToString( const Filename : string) : string
13911:  Function lGetTempFilename( const Directory, Prefix : string) : string
13912:  {Function NeedRTLAnsi : boolean
13913:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13914:  Function UTF8ToSys( const s : string) : string
13915:  Function SysToUTF8( const s : string) : string
13916:  Function ConsoleToUTF8( const s : string) : string
13917:  Function UTF8ToConsole( const s : string) : string}
13918:  Function FileExistsUTF8( const Filename : string) : boolean
13919:  Function FileAgeUTF8( const FileName : string) : Longint
13920:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13921:  Function ExpandFileNameUTF8( const FileName : string) : string
13922:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13923:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13924:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13925:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13926:  Procedure FindCloseUTF8( var F : TSearchrec)
13927:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13928:  Function FileGetAttrUTF8( const FileName : String) : Longint
13929:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13930:  Function DeleteFileUTF8( const FileName : String) : Boolean
13931:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13932:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13933:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13934:  Function GetCurrentDirUTF8 : String
13935:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13936:  Function CreateDirUTF8( const NewDir : String) : Boolean
13937:  Function RemoveDirUTF8( const Dir : String) : Boolean
13938:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13939:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13940:  Function FileCreateUTF8( const FileName : string) : THandle;
13941:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13942:  Function ParamStrUTF8( Param : Integer) : string
13943:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13944:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13945:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13946:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13947:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13948: end;
13949:
13950: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13951: begin
13952:   //VK_F23 = 134;
13953:   //{$EXTERNALSYM VK_F24}
13954:   //VK_F24 = 135;
13955:  TVirtualKeyCode', 'Integer
13956:  'VK_MOUSEWHEELUP','integer'(134);
13957:  'VK_MOUSEWHEELDOWN','integer'(135);
13958:  Function glIsKeyDown( c : Char) : Boolean;
13959:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13960:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
```

```
13961:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13962:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13963:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13964:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13965: end;
13966:
13967: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13968: begin
13969:   TGLPoint', 'TPoint
13970:   //PGLPoint', '^TGLPoint // will not work
13971:   TGLRect', 'TRect
13972:   //PGLRect', '^TGLRect // will not work
13973:   TDelphiColor', 'TColor
13974:   TGLPicture', 'TPicture
13975:   TGLGraphic', 'TGraphic
13976:   TGLBitmap', 'TBitmap
13977:   //TGraphicClass', 'class of TGraphic
13978:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
13979:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
13980:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
13981:    +'Button; Shift : TShiftState; X, Y : Integer)
13982:   TGLMouseMoveEvent', 'TMouseMoveEvent
13983:   TGLKeyEvent', 'TKeyEvent
13984:   TGLKeyPressEvent', 'TKeyPressEvent
13985:   EGLOSError', 'EWin32Error
13986:   EGLOSError', 'EWin32Error
13987:   EGLOSError', 'EOSError
13988:  'glsAllFilter','string'All // sAllFilter
13989:  Function GLPoint( const x, y : Integer) : TGLPoint
13990:  Function GLRGB( const r, g, b : Byte) : TColor
13991:  Function GLColorToRGB( color : TColor) : TColor
13992:  Function GLGetRValue( rgb : DWORD) : Byte
13993:  Function GLGetGValue( rgb : DWORD) : Byte
13994:  Function GLGetBValue( rgb : DWORD) : Byte
13995:  Procedure GLInitWinColors
13996:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
13997:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
13998:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
13999:  Procedure GLInformationDlg( const msg : String)
14000:  Function GLQuestionDlg( const msg : String) : Boolean
14001:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
14002:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
14003:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
14004:  Function GLApplicationTerminated : Boolean
14005:  Procedure GLRaiseLastOSError
14006:  Procedure GLFreeAndNil( var anObject: TObject)
14007:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
14008:  Function GLGetCurrentColorDepth : Integer
14009:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14010:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14011:  Procedure GLSleep( length : Cardinal)
14012:  Procedure GLQueryPerformanceCounter( var val : Int64)
14013:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14014:  Function GLStartPrecisionTimer : Int64
14015:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14016:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14017:  Function GLRDTSC : Int64
14018:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14019:  Function GLOKMessageBox( const Text, Caption : string) : Integer
14020:  Procedure GLShowHTMLUrl( Url : String)
14021:  Procedure GLShowCursor( AShow : boolean)
14022:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14023:  Procedure GLGetCursorPos( var point : TGLPoint)
14024:  Function GLGetScreenWidth : integer
14025:  Function GLGetScreenHeight : integer
14026:  Function GLGetTickCount : int64
14027:  function RemoveSpaces(const str : String) : String;
14028:   TNormalMapSpace','( nmsObject, nmsTangent )
14029:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14030:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14031:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
14032:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
14033:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14034: end;
14035:
14036: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14037: begin
14038:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14039:  // PGLStarRecord', '^TGLStarRecord // will not work
14040:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14041:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14042:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14043: end;
14044:
14045:
14046: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
```

```
14047: begin
14048:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14049:   //PAABB', '^TAABB // will not work
14050:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14051:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14052:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14053:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14054:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14055:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14056:  Procedure SetAABB( var bb : TAABB; const v : TVector)
14057:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14058:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14059:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14060:  Function BBMinX( const c : THmgBoundingBox) : Single
14061:  Function BBMaxX( const c : THmgBoundingBox) : Single
14062:  Function BBMinY( const c : THmgBoundingBox) : Single
14063:  Function BBMaxY( const c : THmgBoundingBox) : Single
14064:  Function BBMinZ( const c : THmgBoundingBox) : Single
14065:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14066:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14067:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14068:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14069:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
14070:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14071:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14072:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14073:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14074:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14075:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14076:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14077:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14078:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14079:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14080:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14081:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14082:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14083:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14084:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14085:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14086:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14087:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14088:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14089:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14090:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14091:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14092:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
       testBSphere:TBSphere):TSpaceContains
14093:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14094:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14095:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14096:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14097:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14098:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
       viewportSizeY:Int):TClipRect
14099: end;
14100:
14101: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14102: begin
14103:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14104:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14105:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14106:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14107:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14108:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14109:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14110:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14111:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14112:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14113:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14114:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14115:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14116:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14117:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14118:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14119:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14120:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14121:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14122:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14123:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14124:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14125:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14126:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14127:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14128: end;
14129:
14130: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14131: begin
14132:  'EPSILON','Single').setExtended( 1e-40);
14133:  'EPSILON2','Single').setExtended( 1e-30);  }
```

```
14134:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14135:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14136:  THmgPlane', 'TVector
14137:   TDoubleHmgPlane', 'THomogeneousDblVector
14138:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14139:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14140:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14141:   TSingleArray', 'array of Single
14142:   TTransformations','array [0..15] of Single)
14143:   TPackedRotationMatrix','array [0..2] of Smallint)
14144:   TVertex', 'TAffineVector
14145:   //TVectorGL', 'THomogeneousFltVector
14146:   //TMatrixGL', 'THomogeneousFltMatrix
14147:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14148:  Function glTexPointMake( const s, t : Single) : TTexPoint
14149:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14150:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14151:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14152:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14153:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14154:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14155:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14156:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14157:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14158:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14159:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14160:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14161:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14162:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14163:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14164:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14165:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14166:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14167:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14168:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14169:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14170:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14171:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14172:  Procedure glRstVector( var v : TAffineVector);
14173:  Procedure glRstVector1( var v : TVectorGL);
14174:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14175:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14176:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14177:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14178:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14179:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14180:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14181:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14182:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14183:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14184:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14185:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14186:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Int;dest:PTexPointArray);
14187:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14188:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
      PAffineVectorArray);
14189:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14190:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14191:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14192:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14193:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14194:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14195:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14196:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14197:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14198:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14199:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14200:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14201:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14202:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14203:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14204:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14205:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14206:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14207:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14208:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14209:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14210:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14211:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14212:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14213:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14214:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14215:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14216:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14217:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14218:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14219:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
```

```
14220:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14221:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14222:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14223:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14224:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14225:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14226:  Function glLerp( const start, stop, t : Single) : Single
14227:  Function glAngleLerp( start, stop, t : Single) : Single
14228:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14229:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14230:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14231:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14232:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14233:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14234:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14235:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14236: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14237: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
       PAffineVectorArray);
14238:  Function glVectorLength( const x, y : Single) : Single;
14239:  Function glVectorLength1( const x, y, z : Single) : Single;
14240:  Function glVectorLength2( const v : TAffineVector) : Single;
14241:  Function glVectorLength3( const v : TVectorGL) : Single;
14242:  Function glVectorLength4( const v : array of Single) : Single;
14243:  Function glVectorNorm( const x, y : Single) : Single;
14244:  Function glVectorNorm1( const v : TAffineVector) : Single;
14245:  Function glVectorNorm2( const v : TVectorGL) : Single;
14246:  Function glVectorNorm3( var V : array of Single) : Single;
14247:  Procedure glNormalizeVector( var v : TAffineVector);
14248:  Procedure glNormalizeVector1( var v : TVectorGL);
14249:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14250:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14251: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14252:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14253:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14254:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14255:  Procedure glNegateVector( var V : TAffineVector);
14256:  Procedure glNegateVector2( var V : TVectorGL);
14257:  Procedure glNegateVector3( var V : array of Single);
14258:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14259:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14260:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14261:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14262:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14263:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14264:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14265:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14266:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14267:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14268:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14269:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14270:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14271:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14272:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14273:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14274:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14275:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14276:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14277:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14278:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14279:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14280:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14281:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14282:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14283:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14284:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14285:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14286:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14287:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14288:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14289:  Procedure glAbsVector( var v : TVectorGL);
14290:  Procedure glAbsVector1( var v : TAffineVector);
14291:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14292:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14293:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14294:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14295:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14296:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14297:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14298:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14299:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14300:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14301:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14302:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14303:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14304:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14305:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14306:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14307:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
```

```
14308:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14309:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14310:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14311:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14312:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14313:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14314:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14315:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14316:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14317:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14318:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14319:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14320:  Procedure glAdjointMatrix( var M : TMatrixGL);
14321:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14322:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14323:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14324:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14325:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14326:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14327:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14328:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14329:  Procedure glInvertMatrix( var M : TMatrixGL);
14330:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14331:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14332:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14333:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14334:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14335:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14336:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14337:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14338:  Procedure glNormalizePlane( var plane : THmgPlane)
14339:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14340:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14341:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14342:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14343:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14344:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14345:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14346:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14347:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14348:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14349:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14350:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14351:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14352:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
      Segment0Closest, Segment1Closest : TAffineVector)
14353:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14354:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14355:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14356:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14357:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14358:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14359:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14360:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14361:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14362:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14363:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14364:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14365:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14366:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14367:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14368:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14369:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14370:  Function glLnXP1( X : Extended) : Extended
14371:  Function glLog10( X : Extended) : Extended
14372:  Function glLog2( X : Extended) : Extended;
14373:  Function glLog21( X : Single) : Single;
14374:  Function glLogN( Base, X : Extended) : Extended
14375:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14376:  Function glPower( const Base, Exponent : Single) : Single;
14377:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14378:  Function glDegToRad( const Degrees : Extended) : Extended;
14379:  Function glDegToRad1( const Degrees : Single) : Single;
14380:  Function glRadToDeg( const Radians : Extended) : Extended;
14381:  Function glRadToDeg1( const Radians : Single) : Single;
14382:  Function glNormalizeAngle( angle : Single) : Single
14383:  Function glNormalizeDegAngle( angle : Single) : Single
14384:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14385:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14386:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14387:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14388:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14389:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14390:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14391:  Function glArcCos( const X : Extended) : Extended;
14392:  Function glArcCos1( const x : Single) : Single;
14393:  Function glArcSin( const X : Extended) : Extended;
14394:  Function glArcSin1( const X : Single) : Single;
14395:  Function glArcTan21( const Y, X : Extended) : Extended;
```

```
14396:  Function glArcTan21( const Y, X : Single) : Single;
14397:  Function glFastArcTan2( y, x : Single) : Single
14398:  Function glTan( const X : Extended) : Extended;
14399:  Function glTan1( const X : Single) : Single;
14400:  Function glCoTan( const X : Extended) : Extended;
14401:  Function glCoTan1( const X : Single) : Single;
14402:  Function glSinh( const x : Single) : Single;
14403:  Function glSinh1( const x : Double) : Double;
14404:  Function glCosh( const x : Single) : Single;
14405:  Function glCosh1( const x : Double) : Double;
14406:  Function glRSqrt( v : Single) : Single
14407:  Function glRLength( x, y : Single) : Single
14408:  Function glISqrt( i : Integer) : Integer
14409:  Function glILength( x, y : Integer) : Integer;
14410:  Function glILength1( x, y, z : Integer) : Integer;
14411:  Procedure glRegisterBasedExp
14412:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14413:  Function glRoundInt( v : Single) : Single;
14414:  Function glRoundInt1( v : Extended) : Extended;
14415:  Function glTrunc( v : Single) : Integer;
14416:  Function glTrunc64( v : Extended) : Int64;
14417:  Function glInt( v : Single) : Single;
14418:  Function glInt1( v : Extended) : Extended;
14419:  Function glFrac( v : Single) : Single;
14420:  Function glFrac1( v : Extended) : Extended;
14421:  Function glRound( v : Single) : Integer;
14422:  Function glRound64( v : Single) : Int64;
14423:  Function glRound641( v : Extended) : Int64;
14424:  Function glTrunc( X : Extended) : Int64
14425:  Function glRound( X : Extended) : Int64
14426:  Function glFrac( X : Extended) : Extended
14427:  Function glCeil( v : Single) : Integer;
14428:  Function glCeil64( v : Extended) : Int64;
14429:  Function glFloor( v : Single) : Integer;
14430:  Function glFloor64( v : Extended) : Int64;
14431:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14432:  Function glSign( x : Single) : Integer
14433:  Function glIsInRange( const x, a, b : Single) : Boolean;
14434:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14435:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14436:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14437:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14438:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14439:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14440:  Function glMinFloat3( const v1, v2 : Single) : Single;
14441:  Function glMinFloat4( const v : array of Single) : Single;
14442:  Function glMinFloat5( const v1, v2 : Double) : Double;
14443:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14444:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14445:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14446:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14447:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14448:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14449:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14450:  Function glMaxFloat2( const v : array of Single) : Single;
14451:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14452:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14453:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14454:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14455:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14456:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14457:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14458:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14459:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14460:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14461:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14462:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14463:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14464:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14465:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14466:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14467:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14468:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14469:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14470:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14471:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14472:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14473:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14474:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14475:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14476:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14477:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14478:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14479:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14480:  Procedure glSortArrayAscending( var a : array of Extended)
14481:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14482:  Function glClampValue1( const aValue, aMin : Single) : Single;
14483:  Function glGeometryOptimizationMode : String
14484:  Procedure glBeginFPUOnlySection
```

```
14485:  Procedure glEndFPUOnlySection
14486:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14487:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14488:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14489:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14490:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14491:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14492:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14493:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14494:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14495:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14496:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14497:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14498:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14499:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14500:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14501:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14502:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14503:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14504:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14505:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14506:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14507:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14508:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14509:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14510:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14511:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14512:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14513:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14514:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14515:  'cPI','Single').setExtended( 3.141592654);
14516:  'cPIdiv180','Single').setExtended( 0.017453292);
14517:  'c180divPI','Single').setExtended( 57.29577951);
14518:  'c2PI','Single').setExtended( 6.283185307);
14519:  'cPIdiv2','Single').setExtended( 1.570796326);
14520:  'cPIdiv4','Single').setExtended( 0.785398163);
14521:  'c3PIdiv4','Single').setExtended( 2.35619449);
14522:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14523:  'cInv360','Single').setExtended( 1 / 360);
14524:  'c180','Single').setExtended( 180);
14525:  'c360','Single').setExtended( 360);
14526:  'cOneHalf','Single').setExtended( 0.5);
14527:  'cLn10','Single').setExtended( 2.302585093);
14528:  {'MinSingle','Extended').setExtended( 1.5e-45);
14529:  'MaxSingle','Extended').setExtended( 3.4e+38);
14530:  'MinDouble','Extended').setExtended( 5.0e-324);
14531:  'MaxDouble','Extended').setExtended( 1.7e+308);
14532:  'MinExtended','Extended').setExtended( 3.4e-4932);
14533:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14534:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14535:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14536: end;
14537:
14538: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14539: begin
14540:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14541:   (FindClass('TOBJECT'),'TFaceGroups
14542:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14543:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14544:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14545:   SIRegister_TBaseMeshObject(CL);
14546:   (FindClass('TOBJECT'),'TSkeletonFrameList
14547:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14548:   SIRegister_TSkeletonFrame(CL);
14549:   SIRegister_TSkeletonFrameList(CL);
14550:   (FindClass('TOBJECT'),'TSkeleton
14551:   (FindClass('TOBJECT'),'TSkeletonBone
14552:   SIRegister_TSkeletonBoneList(CL);
14553:   SIRegister_TSkeletonRootBoneList(CL);
14554:   SIRegister_TSkeletonBone(CL);
14555:   (FindClass('TOBJECT'),'TSkeletonColliderList
14556:   SIRegister_TSkeletonCollider(CL);
14557:   SIRegister_TSkeletonColliderList(CL);
14558:   (FindClass('TOBJECT'),'TGLBaseMesh
14559:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14560:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14561:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14562:    +'QuaternionList; end
14563:   SIRegister_TSkeleton(CL);
14564:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14565:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14566:   SIRegister_TMeshObject(CL);
14567:   SIRegister_TMeshObjectList(CL);
14568:   //TMeshObjectListClass', 'class of TMeshObjectList
```

```
14569:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14570:   SIRegister_TMeshMorphTarget(CL);
14571:   SIRegister_TMeshMorphTargetList(CL);
14572:   SIRegister_TMorphableMeshObject(CL);
14573:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14574:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14575:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14576:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14577:   SIRegister_TSkeletonMeshObject(CL);
14578:   SIRegister_TFaceGroup(CL);
14579:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14580:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14581:   SIRegister_TFGVertexIndexList(CL);
14582:   SIRegister_TFGVertexNormalTexIndexList(CL);
14583:   SIRegister_TFGIndexTexCoordList(CL);
14584:   SIRegister_TFaceGroups(CL);
14585:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14586:   SIRegister_TVectorFile(CL);
14587:   //TVectorFileClass', 'class of TVectorFile
14588:   SIRegister_TGLGLSMVectorFile(CL);
14589:   SIRegister_TGLBaseMesh(CL);
14590:   SIRegister_TGLFreeForm(CL);
14591:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14592:   TGLActorOptions', 'set of TGLActorOption
14593:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14594:   (FindClass('TOBJECT'),'TGLActor
14595:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14596:   SIRegister_TActorAnimation(CL);
14597:   TActorAnimationName', 'String
14598:   SIRegister_TActorAnimations(CL);
14599:   SIRegister_TGLBaseAnimationControler(CL);
14600:   SIRegister_TGLAnimationControler(CL);
14601:   TActorFrameInterpolation', '( afpNone, afpLinear )
14602:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14603:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14604:   SIRegister_TGLActor(CL);
14605:   SIRegister_TVectorFileFormat(CL);
14606:   SIRegister_TVectorFileFormatsList(CL);
14607:   (FindClass('TOBJECT'),'EInvalidVectorFile
14608:  Function GetVectorFileFormats : TVectorFileFormatsList
14609:  Function VectorFileFormatsFilter : String
14610:  Function VectorFileFormatsSaveFilter : String
14611:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14612:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14613:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14614: end;
14615:
14616: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14617: begin
14618:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14619:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14620:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14621:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14622:   SIRegister_TOleStream(CL);
14623:   (FindClass('TOBJECT'),'TConnectionPoints
14624:   TConnectionKind', '( ckSingle, ckMulti )
14625:   SIRegister_TConnectionPoint(CL);
14626:   SIRegister_TConnectionPoints(CL);
14627:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14628:   (FindClass('TOBJECT'),'TActiveXControlFactory
14629:   SIRegister_TActiveXControl(CL);
14630:   //TActiveXControlClass', 'class of TActiveXControl
14631:   SIRegister_TActiveXControlFactory(CL);
14632:   SIRegister_TActiveFormControl(CL);
14633:   SIRegister_TActiveForm(CL);
14634:   //TActiveFormClass', 'class of TActiveForm
14635:   SIRegister_TActiveFormFactory(CL);
14636:   (FindClass('TOBJECT'),'TPropertyPageImpl
14637:   SIRegister_TPropertyPage(CL);
14638:   //TPropertyPageClass', 'class of TPropertyPage
14639:   SIRegister_TPropertyPageImpl(CL);
14640:   SIRegister_TActiveXPropertyPage(CL);
14641:   SIRegister_TActiveXPropertyPageFactory(CL);
14642:   SIRegister_TCustomAdapter(CL);
14643:   SIRegister_TAdapterNotifier(CL);
14644:   SIRegister_IFontAccess(CL);
14645:   SIRegister_TFontAdapter(CL);
14646:   SIRegister_IPictureAccess(CL);
14647:   SIRegister_TPictureAdapter(CL);
14648:   SIRegister_TOleGraphic(CL);
14649:   SIRegister_TStringsAdapter(CL);
14650:   SIRegister_TReflectorWindow(CL);
14651:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14652:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14653:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14654:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14655:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14656:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14657:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
```

```
14658:  Function ParkingWindow : HWND
14659: end;
14660:
14661: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14662: begin
14663:  // TIp6Bytes = array [0..15] of Byte;
14664: {:binary form of IPv6 adress (for string conversion routines)}
14665:  // TIp6Words = array [0..7] of Word;
14666:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14667:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14668:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14669:  Function synaIsIP( const Value : string) : Boolean');
14670:  Function synaIsIP6( const Value : string) : Boolean');
14671:  Function synaIPToID( Host : string) : Ansistring');
14672:  Function synaStrToIp6( value : string) : TIp6Bytes');
14673:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14674:  Function synaStrToIp( value : string) : integer');
14675:  Function synaIpToStr( value : integer) : string');
14676:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14677:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14678:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14679:  Function xStrToIP( const Value : String) : TIPAdr');
14680:  Function xIPToStr( const Adresse : TIPAdr) : String');
14681:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14682:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14683: end;
14684:
14685: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14686: begin
14687:  AddTypeS('TSpecials', 'set of Char');
14688:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14689:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14690:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14691:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14692:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14693:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14694:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14695:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14696:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14697:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14698:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14699:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14700:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
14701:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14702:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14703:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14704:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
14705:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14706:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14707:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14708:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14709:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14710:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14711:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14712:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14713:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
14714:  Function synCrc32( const Value : AnsiString) : Integer');
14715:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14716:  Function Crc16( const Value : AnsiString) : Word');
14717:  Function synMD5( const Value : AnsiString) : AnsiString');
14718:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14719:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14720:  Function synSHA1( const Value : AnsiString) : AnsiString');
14721:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
14722:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14723:  Function synMD4( const Value : AnsiString) : AnsiString');
14724: end;
14725:
14726: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14727: begin
14728:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14729:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14730:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14731:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14732:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14733:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14734:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14735:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14736:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14737:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14738:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14739:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14740:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14741:    +', CP864, CP865, CP869, CP1125 )');
14742:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14743:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14744:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
      TransformTable : array of Word) : AnsiString');
14745:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
      TransformTable : array of Word; Translit : Boolean) : AnsiString');
```

```
14746:  Function GetCurCP : TMimeChar');
14747:  Function GetCurOEMCP : TMimeChar');
14748:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14749:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14750:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14751:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14752:  Function GetBOM( Value : TMimeChar) : AnsiString');
14753:  Function StringToWide( const Value : AnsiString) : WideString');
14754:  Function WideToString( const Value : WideString) : AnsiString');
14755: end;
14756:
14757: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14758: begin
14759:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14760:  Procedure WakeOnLan( MAC, IP : string)');
14761:  Function GetDNS : string');
14762:  Function GetIEProxy( protocol : string) : TProxySetting');
14763:  Function GetLocalIPs : string');
14764: end;
14765:
14766:
14767: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14768: begin
14769:  AddConstantN('synCR','Char #$0d);
14770:  Const('synLF','Char #$0a);
14771:  Const('cSerialChunk','LongInt'( 8192);
14772:  Const('LockfileDirectory','String '/var/lock');
14773:  Const('PortIsClosed','LongInt'( - 1);
14774:  Const('ErrAlreadyOwned','LongInt'( 9991);
14775:  Const('ErrAlreadyInUse','LongInt'( 9992);
14776:  Const('ErrWrongParameter','LongInt'( 9993);
14777:  Const('ErrPortNotOpen','LongInt'( 9994);
14778:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14779:  Const('ErrMaxBuffer','LongInt'( 9996);
14780:  Const('ErrTimeout','LongInt'( 9997);
14781:  Const('ErrNotRead','LongInt'( 9998);
14782:  Const('ErrFrame','LongInt'( 9999);
14783:  Const('ErrOverrun','LongInt'( 10000);
14784:  Const('ErrRxOver','LongInt'( 10001);
14785:  Const('ErrRxParity','LongInt'( 10002);
14786:  Const('ErrTxFull','LongInt'( 10003);
14787:  Const('dcb_Binary','LongWord')( $00000001);
14788:  Const('dcb_ParityCheck','LongWord')( $00000002);
14789:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14790:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14791:  Const('dcb_DtrControlMask','LongWord')( $00000030);
14792:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14793:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14794:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14795:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14796:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14797:  Const('dcb_OutX','LongWord')( $00000100);
14798:  Const('dcb_InX','LongWord')( $00000200);
14799:  Const('dcb_ErrorChar','LongWord')( $00000400);
14800:  Const('dcb_NullStrip','LongWord')( $00000800);
14801:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14802:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14803:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14804:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14805:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14806:  Const('dcb_AbortOnError','LongWord')( $00004000);
14807:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14808:  Const('synSB1','LongInt'( 0);
14809:  Const('SB1andHalf','LongInt'( 1);
14810:  Const('synSB2','LongInt'( 2);
14811:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14812:  Const('CS7fix','LongWord')( $0000020);
14813:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14814:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14815:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14816:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14817:   //AddTypeS('PDCB', '^TDCB // will not work');
14818:  //Const('MaxRates','LongInt'( 18);
14819:  //Const('MaxRates','LongInt'( 30);
14820:  //Const('MaxRates','LongInt'( 19);
14821:  Const('O_SYNC','LongWord')( $0080);
14822:  Const('synOK','LongInt'( 0);
14823:  Const('synErr','LongInt'( integer ( - 1 ));
14824:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
      HR_WriteCount, HR_Wait )');
14825:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14826:   SIRegister_ESynaSerError(CL);
14827:   SIRegister_TBlockSerial(CL);
14828:  Function GetSerialPortNames : string');
14829: end;
14830:
14831: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14832: begin
14833:  Const('DLLIconvName','String 'libiconv.so');
```

```
14834:  Const('DLLIconvName','String 'iconv.dll');
14835:   AddTypeS('size_t', 'Cardinal');
14836:   AddTypeS('iconv_t', 'Integer');
14837:   //AddTypeS('iconv_t', 'Pointer');
14838:   AddTypeS('argptr', 'iconv_t');
14839:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14840:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14841:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14842:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14843:  Function SynaIconvClose( var cd : iconv_t) : integer');
14844:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14845:  Function IsIconvloaded : Boolean');
14846:  Function InitIconvInterface : Boolean');
14847:  Function DestroyIconvInterface : Boolean');
14848:  Const('ICONV_TRIVIALP','LongInt'( 0);
14849:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14850:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14851:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14852:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14853: end;
14854:
14855: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14856: begin
14857:  Const 'ICMP_ECHO','LongInt'( 8);
14858:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14859:  Const('ICMP_UNREACH','LongInt'( 3);
14860:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14861:  Const('ICMP6_ECHO','LongInt'( 128);
14862:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14863:  Const('ICMP6_UNREACH','LongInt'( 1);
14864:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14865:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14866:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14867:   SIRegister_TPINGSend(CL);
14868:  Function PingHost( const Host : string) : Integer');
14869:  Function TraceRouteHost( const Host : string) : string');
14870: end;
14871:
14872: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14873: begin
14874:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14875:  Const('synASN1_INT','LongWord')( $02);
14876:  Const('synASN1_OCTSTR','LongWord')( $04);
14877:  Const('synASN1_NULL','LongWord')( $05);
14878:  Const('synASN1_OBJID','LongWord')( $06);
14879:  Const('synASN1_ENUM','LongWord')( $0a);
14880:  Const('synASN1_SEQ','LongWord')( $30);
14881:  Const('synASN1_SETOF','LongWord')( $31);
14882:  Const('synASN1_IPADDR','LongWord')( $40);
14883:  Const('synASN1_COUNTER','LongWord')( $41);
14884:  Const('synASN1_GAUGE','LongWord')( $42);
14885:  Const('synASN1_TIMETICKS','LongWord')( $43);
14886:  Const('synASN1_OPAQUE','LongWord')( $44);
14887:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14888:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
14889:  Function synASNEncLen( Len : Integer) : AnsiString');
14890:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14891:  Function synASNEncInt( Value : Integer) : AnsiString');
14892:  Function synASNEncUInt( Value : Integer) : AnsiString');
14893:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14894:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14895:  Function synMibToId( Mib : String) : AnsiString');
14896:  Function synIdToMib( const Id : AnsiString) : String');
14897:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14898:  Function ASNdump( const Value : AnsiString) : AnsiString');
14899:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
14900:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14901: end;
14902:
14903: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14904: begin
14905:  Const('cLDAPProtocol','String '389');
14906:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14907:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14908:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14909:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14910:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14911:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14912:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14913:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14914:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14915:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14916:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14917:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14918:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14919:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14920:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14921:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14922:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
```

```
14923:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14924:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14925:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14926:   SIRegister_TLDAPAttribute(CL);
14927:   SIRegister_TLDAPAttributeList(CL);
14928:   SIRegister_TLDAPResult(CL);
14929:   SIRegister_TLDAPResultList(CL);
14930:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14931:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14932:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14933:   SIRegister_TLDAPSend(CL);
14934:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14935: end;
14936:
14937:
14938: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14939: begin
14940:  Const('cSysLogProtocol','String '514');
14941:  Const('FCL_Kernel','LongInt'( 0);
14942:  Const('FCL_UserLevel','LongInt'( 1);
14943:  Const('FCL_MailSystem','LongInt'( 2);
14944:  Const('FCL_System','LongInt'( 3);
14945:  Const('FCL_Security','LongInt'( 4);
14946:  Const('FCL_Syslogd','LongInt'( 5);
14947:  Const('FCL_Printer','LongInt'( 6);
14948:  Const('FCL_News','LongInt'( 7);
14949:  Const('FCL_UUCP','LongInt'( 8);
14950:  Const('FCL_Clock','LongInt'( 9);
14951:  Const('FCL_Authorization','LongInt'( 10);
14952:  Const('FCL_FTP','LongInt'( 11);
14953:  Const('FCL_NTP','LongInt'( 12);
14954:  Const('FCL_LogAudit','LongInt'( 13);
14955:  Const('FCL_LogAlert','LongInt'( 14);
14956:  Const('FCL_Time','LongInt'( 15);
14957:  Const('FCL_Local0','LongInt'( 16);
14958:  Const('FCL_Local1','LongInt'( 17);
14959:  Const('FCL_Local2','LongInt'( 18);
14960:  Const('FCL_Local3','LongInt'( 19);
14961:  Const('FCL_Local4','LongInt'( 20);
14962:  Const('FCL_Local5','LongInt'( 21);
14963:  Const('FCL_Local6','LongInt'( 22);
14964:  Const('FCL_Local7','LongInt'( 23);
14965:   Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
14966:   SIRegister_TSyslogMessage(CL);
14967:   SIRegister_TSyslogSend(CL);
14968:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
      Content:string):Boolean;
14969: end;
14970:
14971:
14972: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14973: begin
14974:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14975:   SIRegister_TMessHeader(CL);
14976:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14977:   SIRegister_TMimeMess(CL);
14978: end;
14979:
14980: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14981: begin
14982:   (FindClass('TOBJECT'),'TMimePart');
14983:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
14984:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14985:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14986:   SIRegister_TMimePart(CL);
14987:  Const('MaxMimeType','LongInt'( 25);
14988:  Function GenerateBoundary : string');
14989: end;
14990:
14991: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
14992: begin
14993:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
14994:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
14995:  Function NeedInline( const Value : AnsiString) : boolean');
14996:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
14997:  Function InlineCode( const Value : string) : string');
14998:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
14999:  Function InlineEmail( const Value : string) : string');
15000: end;
15001:
15002: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
15003: begin
15004:  Const('cFtpProtocol','String '21');
15005:  Const('cFtpDataProtocol','String '20');
15006:  Const('FTP_OK','LongInt'( 255);
15007:  Const('FTP_ERR','LongInt'( 254);
15008:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15009:   SIRegister_TFTPListRec(CL);
15010:   SIRegister_TFTPList(CL);
```

```
15011:   SIRegister_TFTPSend(CL);
15012:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15013:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15014:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
        ToPort, ToFile, ToUser, ToPass : string) : Boolean');
15015: end;
15016:
15017: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
15018: begin
15019:  Const('cHttpProtocol','String '80');
15020:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15021:   SIRegister_THTTPSend(CL);
15022:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
15023:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
15024:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
15025:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
15026:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
        ResultData:TStrings):Bool;
15027: end;
15028:
15029: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15030: begin
15031:  Const('cSmtpProtocol','String '25');
15032:   SIRegister_TSMTPSend(CL);
15033:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
        Passw:string):Bool;
15034:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15035:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
        Username, Password : string):Boolean');
15036: end;
15037:
15038: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15039: begin
15040:  Const('cSnmpProtocol','String '161');
15041:  Const('cSnmpTrapProtocol','String '162');
15042:  Const('SNMP_V1','LongInt'( 0);
15043:  Const('SNMP_V2C','LongInt'( 1);
15044:  Const('SNMP_V3','LongInt'( 3);
15045:  Const('PDUGetRequest','LongWord')( $A0);
15046:  Const('PDUGetNextRequest','LongWord')( $A1);
15047:  Const('PDUGetResponse','LongWord')( $A2);
15048:  Const('PDUSetRequest','LongWord')( $A3);
15049:  Const('PDUTrap','LongWord')( $A4);
15050:  Const('PDUGetBulkRequest','LongWord')( $A5);
15051:  Const('PDUInformRequest','LongWord')( $A6);
15052:  Const('PDUTrapV2','LongWord')( $A7);
15053:  Const('PDUReport','LongWord')( $A8);
15054:  Const('ENoError',LongInt 0;
15055:  Const('ETooBig','LongInt')( 1);
15056:  Const('ENoSuchName','LongInt'( 2);
15057:  Const('EBadValue','LongInt'( 3);
15058:  Const('EReadOnly','LongInt'( 4);
15059:  Const('EGenErr','LongInt'( 5);
15060:  Const('ENoAccess','LongInt'( 6);
15061:  Const('EWrongType','LongInt'( 7);
15062:  Const('EWrongLength','LongInt'( 8);
15063:  Const('EWrongEncoding','LongInt'( 9);
15064:  Const('EWrongValue','LongInt'( 10);
15065:  Const('ENoCreation','LongInt'( 11);
15066:  Const('EInconsistentValue','LongInt'( 12);
15067:  Const('EResourceUnavailable','LongInt'( 13);
15068:  Const('ECommitFailed','LongInt'( 14);
15069:  Const('EUndoFailed','LongInt'( 15);
15070:  Const('EAuthorizationError','LongInt'( 16);
15071:  Const('ENotWritable','LongInt'( 17);
15072:  Const('EInconsistentName','LongInt'( 18);
15073:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15074:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15075:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15076:   SIRegister_TSNMPMib(CL);
15077:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15078:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15079:   SIRegister_TSNMPRec(CL);
15080:   SIRegister_TSNMPSend(CL);
15081:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15082:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
15083:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15084:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15085:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15086:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
        Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15087:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
        Integer; const MIBName, MIBValue : TStringList) : Integer');
15088: end;
15089:
15090: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15091: begin
15092:  Function GetDomainName2: AnsiString');
15093:  Function GetDomainController( Domain : AnsiString) : AnsiString');
```

```
15094:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15095:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15096:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15097:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15098: end;
15099:
15100: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15101: begin
15102:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15103:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15104:  Function wwStrToDate( const S : string) : boolean');
15105:  Function wwStrToTime( const S : string) : boolean');
15106:  Function wwStrToDateTime( const S : string) : boolean');
15107:  Function wwStrToTimeVal( const S : string) : TDateTime');
15108:  Function wwStrToDateVal( const S : string) : TDateTime');
15109:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15110:  Function wwStrToInt( const S : string) : boolean');
15111:  Function wwStrToFloat( const S : string) : boolean');
15112:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15113:  Function wwNextDay( Year, Month, Day : Word) : integer');
15114:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15115:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15116:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15117:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15118:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15119:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15120:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15121:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15122:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15123: end;
15124:
15125: unit uPSI_Themes;
15126: Function ThemeServices : TThemeServices');
15127: Function ThemeControl( AControl : TControl) : Boolean');
15128: Function UnthemedDesigner( AControl : TControl) : Boolean');
15129: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15130: begin
15131:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15132: end;
15133: Unit uPSC_menus;
15134:  Function StripHotkey( const Text : string) : string');
15135:  Function GetHotkey( const Text : string) : string');
15136:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15137:  Function IsAltGRPressed : boolean');
15138:
15139: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15140: begin
15141:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15142:   SIRegister_TIdIMAP4Server(CL);
15143: end;
15144:
15145: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15146: begin
15147:  'HASH_SIZE','LongInt'( 256);
15148:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15149:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15150:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15151:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15152:    +' : Integer; Value : Variant; end');
15153:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15154:   SIRegister_TVariantSymbolTable(CL);
15155: end;
15156:
15157: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15158: begin
15159:   SIRegister_TThreadLocalVariables(CL);
15160:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15161:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15162:  Function ThreadLocals : TThreadLocalVariables');
15163:  Procedure WriteDebug( sz : String)');
15164:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15165:  'UDF_FAILURE','LongInt'( 1);
15166:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15167:  CL.AddTypeS('mTByteArray', 'array of byte;');
15168:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15169:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15170:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15171:  function IsNetworkConnected: Boolean;
15172:  function IsInternetConnected: Boolean;
15173:  function IsCOMConnected: Boolean;
15174:  function IsNetworkOn: Boolean;
15175:  function IsInternetOn: Boolean;
15176:  function IsCOMOn: Boolean;
15177:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15178:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15179:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15180:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15181:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15182:  Function GetMenu( hWnd : HWND) : HMENU');
```

```
15183:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15184: end;
15185:
15186: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15187: begin
15188:   SIRegister_IDataBlock(CL);
15189:   SIRegister_ISendDataBlock(CL);
15190:   SIRegister_ITransport(CL);
15191:   SIRegister_TDataBlock(CL);
15192:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15193:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15194:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15195:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15196:   SIRegister_TCustomDataBlockInterpreter(CL);
15197:   SIRegister_TSendDataBlock(CL);
15198:   'CallSig','LongWord')( $D800);
15199:   'ResultSig','LongWord')( $D400);
15200:   'asMask','LongWord')( $00FF);
15201:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15202:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15203:  Procedure CheckSignature( Sig : Integer)');
15204: end;
15205:
15206: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15207: begin
15208:   //CL.AddTypeS('HINTERNET', '___Pointer');
15209:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15210:   CL.AddTypeS('HINTERNET', 'Integer');
15211:   CL.AddTypeS('HINTERNET2', '___Pointer');
15212:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15213:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15214:   CL.AddTypeS('INTERNET_PORT', 'Word');
15215:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15216:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15217:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15218:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15219:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15220:  Function InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var
       lpUrlComponents:TURLComponents):BOOL;
15221:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
       dwUrlLength:DWORD):BOOL;
15222:  Function InternetCloseHandle(hInet : HINTERNET) : BOOL');
15223:  Function
       InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
       lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15224:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15225:  ;dwContext:DWORD):HINTERNET;
15226:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
       lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15227:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
       dwContext:DWORD):BOOL;
15228:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15229:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15230:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15231:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15232:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15233:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15234:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15235:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
       lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15236:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
       lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15237:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15238:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15239:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
       dwContext:DWORD):BOOL;
15240:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
       TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15241:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
       BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
15242:  Function
       WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15243:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
15244:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15245:  Function
       FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15246:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15247:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15248:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15249:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15250:  Function
       FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15251:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15252:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15253:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15254:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15255:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15256:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
```

```
15257:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15258:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15259:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15260:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15261:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15262:  Function
        GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
        PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15263:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15264:  Function
        GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15265:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
        PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15266:  Function
        HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15267:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
        dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15268:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15269:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15270:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15271:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15272:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15273:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15274:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15275:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
        lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15276:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
        lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15277:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15278:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15279:  Function
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15280:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15281: end;
15282:
15283: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15284: begin
15285:   AddTypeS('strCharSet', 'set of char');
15286:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15287:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15288:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15289:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
15290:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15291:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15292:  Procedure strStripWhiteSpace( var s : string)');
15293:  Function strRemoveChar( str : string; removeChar : char) : string');
15294:  Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15295:  Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15296:  Function wwEqualStr( s1, s2 : string) : boolean');
15297:  Function strCount( s : string; delimeter : char) : integer');
15298:  Function strWhiteSpace : strCharSet');
15299:  Function wwExtractFileNameOnly( const FileName : string) : string');
15300:  Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15301:  Function strTrailing( s : string; delimeter : char) : string');
15302:  Function strPreceding( s : string; delimeter : char) : string');
15303:  Function wwstrReplace( s, Find, Replace : string) : string');
15304: end;
15305:
15306: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15307: begin
15308:   SIRegister_TRemoteDataModule(CL);
15309:   SIRegister_TCRemoteDataModule(CL);
15310:  Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)');
15311:  Procedure UnregisterPooled( const ClassID : string)');
15312:  Procedure EnableSocketTransport( const ClassID : string)');
15313:  Procedure DisableSocketTransport( const ClassID : string)');
15314:  Procedure EnableWebTransport( const ClassID : string)');
15315:  Procedure DisableWebTransport( const ClassID : string)');
15316: end;
15317:
15318: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15319: begin
15320:  Function mxArcCos( x : Real) : Real');
15321:  Function mxArcSin( x : Real) : Real');
15322:  Function Comp2Str( N : Comp) : String');
15323:  Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15324:  Function Int2Str( N : LongInt) : String');
15325:  Function mxIsEqual( R1, R2 : Double) : Boolean');
15326:  Function LogXY( x, y : Real) : Real');
15327:  Function Pennies2Dollars( C : Comp) : String');
15328:  Function mxPower( X : Integer; Y : Integer) : Real');
15329:  Function Real2Str( N : Real; Width, Places : integer) : String');
15330:  Function mxStr2Comp( MyString : string) : Comp');
15331:  Function mxStr2Pennies( S : String) : Comp');
15332:  Function Str2Real( MyString : string) : Real');
15333:  Function XToTheY( x, y : Real) : Real');
15334: end;
15335:
15336: //************************Cindy Functions!*****************************
```

```
15337: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15338: begin
15339:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15340:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15341:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15342:  MessagePlainText','String 'text/plain');
15343:  CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15344:  MessageAlterText_Html','String 'multipart/alternative');
15345:  MessageHtml_Attach','String 'multipart/mixed');
15346:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15347:  MessageAlterText_Html_Attach','String 'multipart/mixed');
15348:  MessageAlterText_Html_RelatedAttach','String')('multipart/related;type="multipart/alternative"';
15349:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15350:  MessageReadNotification','String').( 'multipart/report; report-type="disposition-notification"';
15351:  Function ForceDecodeHeader( aHeader : String) : String');
15352:  Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string');
15353:  Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;');
15354:  Function Base64_DecodeToBytes( Value : String) : TBytes;');
15355:  Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15356:  Function Get_MD5( const aFileName : string) : string');
15357:  Function Get_MD5FromString( const aString : string) : string');
15358: end;
15359:
15360: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15361: begin
15362:  Function IsFolder( SRec : TSearchrec) : Boolean');
15363:  Function isFolderReadOnly( Directory : String) : Boolean');
15364:  Function DirectoryIsEmpty( Directory : String) : Boolean');
15365:  Function DirectoryWithSubDir( Directory : String) : Boolean');
15366:  Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15367:  Function DiskFreeBytes( Drv : Char) : Int64');
15368:  Function DiskBytes( Drv : Char) : Int64');
15369:  Function GetFileBytes( Filename : String) : Int64');
15370:  Function GetFilesBytes( Directory, Filter : String) : Int64');
15371:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15372:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15373:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15374:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15375:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15376:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15377:  SE_TCB_NAME','String 'SeTcbPrivilege');
15378:  SE_SECURITY_NAME','String 'SeSecurityPrivilege');
15379:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15380:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15381:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
15382:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15383:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15384:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15385:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15386:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15387:  SE_BACKUP_NAME','String 'SeBackupPrivilege');
15388:  SE_RESTORE_NAME','String 'SeRestorePrivilege');
15389:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15390:  SE_DEBUG_NAME','String 'SeDebugPrivilege');
15391:  SE_AUDIT_NAME','String 'SeAuditPrivilege');
15392:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15393:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15394:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15395:  SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15396:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15397:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
15398:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15399: end;
15400:
15401:
15402: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15403: begin
15404:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15405:    +'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15406:  Function ShellGetExtensionName( FileName : String) : String');
15407:  Function ShellGetIconIndex( FileName : String) : Integer');
15408:  Function ShellGetIconHandle( FileName : String) : HIcon');
15409:  Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15410:  Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
15411:  Procedure ShellRenameDir( DirFrom, DirTo : string)');
15412:  Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15413:  Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15414:  Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)');
15415:  Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)');
15416:  Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
      WaitCloseCompletion : boolean) : Boolean');
15417:  Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15418:  Function RemoveDuplicatedPathDelimiter( Str : String) : String');
15419:  Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime');
15420:  Function GetModificationDate( Filename : String) : TDateTime');
15421:  Function GetCreationDate( Filename : String) : TDateTime');
15422:  Function GetLastAccessDate( Filename : String) : TDateTime');
15423:  Function FileDelete( Filename : String) : Boolean');
15424:  Function FileIsOpen( Filename : string) : boolean');
```

```
15425:  Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15426:  Function DirectoryDelete( Directory : String) : Boolean');
15427:  Function GetPrinters( PrintersList : TStrings) : Integer');
15428:  Procedure SetDefaultPrinter( PrinterName : String)');
15429:  Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15430:  Function WinToDosPath( WinPathName : String) : String');
15431:  Function DosToWinPath( DosPathName : String) : String');
15432:  Function cyGetWindowsVersion : TWindowsVersion');
15433:  Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean');
15434:  Procedure WindowsShutDown( Restart : boolean)');
15435:  Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
       FileIcon:string;NumIcone:integer);
15436:  Procedure GetWindowsFonts( FontsList : TStrings)');
15437:  Function GetAvailableFilename( DesiredFileName : String): String');
15438: end;
15439:
15440: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15441: begin
15442:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15443:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15444:   Type(TStringRead', '( srFromLeft, srFromRight )');
15445:   Type(TStringReads', 'set of TStringRead');
15446:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15447:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15448:   Type(TWordsOptions', 'set of TWordsOption');
15449:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15450:   Type(TCarTypes', 'set of TCarType');
15451:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15452:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15453:  Function Char_GetType( aChar : Char) : TCarType');
15454:  Function SubString_Count( Str : String; Separator : Char) : Integer');
15455:  Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15456:  Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String');
15457:  Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15458:  Procedure SubString_Add( var Str : String; Separator : Char; Value : String)');
15459:  Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String)');
15460:  Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String)');
15461:  Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean');
15462:  Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15463:  Function SubString_Ribbon( Str : string; Separator : Char; Current: Word; MoveBy : Integer):Integer;');
15464:  Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String;');
15465:  Function String_Quote( Str : String) : String');
15466:  Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char');
15467:  Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15468:  Function String_GetWord( Str : String; StringRead : TStringRead) : String');
15469:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
15470:  Function String_ToInt( Str : String) : Integer');
15471:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15472:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15473:  Function String_Reverse( Str : String) : String');
15474:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15475:  Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
       TCaseSensitive):Integer;');
15476:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15477:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15478:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
       Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15479:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15480:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15481:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15482:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15483:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15484:  Function String_End( Str : String; Cars : Word) : String');
15485:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
       AlwaysFindFromBeginning : Boolean) : String');
15486:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15487:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15488:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15489:  Function String_IsNumbers( Str : String) : Boolean');
15490:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15491:  Function StringToCsvCell( aStr : String) : String');
15492: end;
15493:
15494: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15495: begin
15496:  Function LongDayName( aDate : TDate) : String');
15497:  Function LongMonthName( aDate : TDate) : String');
15498:  Function ShortYearOf( aDate : TDate) : byte');
15499:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15500:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15501:  Function SecondsToMinutes( Seconds : Integer) : Double');
15502:  Function MinutesToSeconds( Minutes : Double) : Integer');
15503:  Function MinutesToHours( Minutes : Integer) : Double');
15504:  Function HoursToMinutes( Hours : Double) : Integer');
15505:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15506:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15507:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15508:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15509:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
```

```
15510:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15511:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
       RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15512:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15513:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15514: end;
15515:
15516: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15517: begin
15518:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15519:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15520:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15521:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15522:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15523:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15524:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15525:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15526:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15527:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15528:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15529:  Function
       TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
15530:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15531:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
       CopySubChildren:Bool;
15532:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15533:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15534:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15535:  Procedure cyCenterControl( aControl : TControl)');
15536:  Function GetLastParent( aControl : TControl) : TWinControl');
15537:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15538:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15539: end;
15540:
15541: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15542: begin
15543:  Function TablePackTable( Tab : TTable) : Boolean');
15544:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15545:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15546:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15547:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15548:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
15549:  Function TableEmptyTable( Tab : TTable) : Boolean');
15550:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15551:  Procedure TableFindNearest( aTable : TTable; Value : String)');
15552:  Function
       TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Boolean):TTable;
15553:  Function
       TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15554:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15555: end;
15556:
15557: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15558: begin
15559:   SIRegister_TcyRunTimeDesign(CL);
15560:   SIRegister_TcyShadowText(CL);
15561:   SIRegister_TcyBgPicture(CL);
15562:   SIRegister_TcyGradient(CL);
15563:   SIRegister_tcyBevel(CL);
15564:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15565:   SIRegister_tcyBevels(CL);
15566:   SIRegister_TcyImagelistOptions(CL);
15567:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15568: end;
15569:
15570: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15571: begin
15572:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
       adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
       Maxdegrade : Byte;'+
15573:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
15574:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15575:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15576:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
15577:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15578:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15579:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15580:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
       BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15581:  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;
15582:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15583:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15584:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
       TColor; aState : TButtonState; Focused, Hot : Boolean)');
```

```
15585:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
        GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15586:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
        const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15587:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
        TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15588:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
        TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15589:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
        WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15590:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15591:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15592:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15593:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
        CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15594:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15595:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15596:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
15597:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15598:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15599:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15600:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15601:  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);
15602:  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;
15603:  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);');
15604:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15605:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15606:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15607:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15608:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
15609:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15610:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15611:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15612:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
15613:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15614:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
15615:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15616:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15617:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15618:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15619:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15620: end;
15621:
15622: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15623: begin
15624:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15625:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15626:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15627:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15628:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15629:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15630:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15631:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15632:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15633:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15634:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15635:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
        bmInvertReverseFromColor);
15636:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
        rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15637:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15638:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15639: end;
15640:
15641: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15642: begin
15643:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15644:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15645:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15646:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15647:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15648:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15649:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15650:  Const SC_GROUP_IDENTIFIERW','String') '+');
15651:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15652:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15653:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15654:  Const SERVICE_INACTIVE','LongWord $00000002);
15655:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
```

```
15656:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15657:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15658:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15659:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15660:  Const SERVICE_STOPPED','LongWord $00000001);
15661:  Const SERVICE_START_PENDING','LongWord $00000002);
15662:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15663:  Const SERVICE_RUNNING','LongWord $00000004);
15664:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15665:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15666:  Const SERVICE_PAUSED','LongWord $00000007);
15667:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15668:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15669:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15670:  Const SC_MANAGER_CONNECT','LongWord $0001);
15671:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15672:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15673:  Const SC_MANAGER_LOCK','LongWord $0008);
15674:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15675:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
15676:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15677:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15678:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15679:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15680:  Const SERVICE_START','LongWord $0010);
15681:  Const SERVICE_STOP','LongWord $0020);
15682:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15683:  Const SERVICE_INTERROGATE','LongWord $0080);
15684:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15685:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15686:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15687:  Const SERVICE_ADAPTER','LongWord $00000004);
15688:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15689:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15690:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15691:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15692:  Const SERVICE_BOOT_START','LongWord $00000000);
15693:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15694:  Const SERVICE_AUTO_START','LongWord $00000002);
15695:  Const SERVICE_DEMAND_START','LongWord $00000003);
15696:  Const SERVICE_DISABLED','LongWord $00000004);
15697:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15698:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15699:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
15700:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15701:   CL.AddTypeS('SC_HANDLE', 'THandle');
15702:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15703:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15704:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15705:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15706:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15707:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15708:   Const TServiceStatus', '_SERVICE_STATUS');
15709:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15710:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15711:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15712:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15713:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15714:   TEnumServiceStatus', 'TEnumServiceStatusA');
15715:   SC_LOCK', '___Pointer');
15716:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15717:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15718:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15719:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15720:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15721:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15722:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15723:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15724:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15725:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15726:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15727:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15728:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15729:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15730:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15731:   TQueryServiceConfig', 'TQueryServiceConfigA');
15732:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15733:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15734:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
       dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15735:  +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15736:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
       dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15737:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15738:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15739:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
       TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15740:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
       lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
       : DWORD):BOOL');
```

```
15741:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
        lpcchBuffer:DWORD):BOOL');
15742:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
        lpcchBuffer:DWORD):BOOL;
15743:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15744:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15745:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15746:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15747:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
        cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15748:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15749:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15750:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15751:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15752: end;
15753:
15754: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15755: begin
15756:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
        AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
        BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15757:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
        DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15758:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15759:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
        TWinControl;
15760:  Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
        AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15761:  Function CreateNotifyThread(const
        FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15762: end;
15763:
15764: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15765: begin
15766:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15767:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15768:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15769:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15770:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
15771:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15772:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15773:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15774:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
15775:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15776:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
15777:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15778:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15779:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15780:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15781:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15782:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15783:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
15784:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15785:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15786:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15787:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15788:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15789:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15790:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15791:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15792:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15793:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15794:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
15795:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15796:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15797:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15798:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15799:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15800:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15801:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15802:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15803:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15804:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
15805:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15806:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15807:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15808:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15809:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15810:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15811:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
        List:TStrings):Bool
15812:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15813:   FindClass('TOBJECT'),'EJclFileSummaryError');
15814:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15815:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15816:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15817:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15818:   SIRegister_TJclFilePropertySet(CL);
15819:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
```

```
15820:   SIRegister_TJclFileSummary(CL);
15821:   SIRegister_TJclFileSummaryInformation(CL);
15822:   SIRegister_TJclDocSummaryInformation(CL);
15823:   SIRegister_TJclMediaFileSummaryInformation(CL);
15824:   SIRegister_TJclMSISummaryInformation(CL);
15825:   SIRegister_TJclShellSummaryInformation(CL);
15826:   SIRegister_TJclStorageSummaryInformation(CL);
15827:   SIRegister_TJclImageSummaryInformation(CL);
15828:   SIRegister_TJclDisplacedSummaryInformation(CL);
15829:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15830:   SIRegister_TJclMiscSummaryInformation(CL);
15831:   SIRegister_TJclWebViewSummaryInformation(CL);
15832:   SIRegister_TJclMusicSummaryInformation(CL);
15833:   SIRegister_TJclDRMSummaryInformation(CL);
15834:   SIRegister_TJclVideoSummaryInformation(CL);
15835:   SIRegister_TJclAudioSummaryInformation(CL);
15836:   SIRegister_TJclControlPanelSummaryInformation(CL);
15837:   SIRegister_TJclVolumeSummaryInformation(CL);
15838:   SIRegister_TJclShareSummaryInformation(CL);
15839:   SIRegister_TJclLinkSummaryInformation(CL);
15840:   SIRegister_TJclQuerySummaryInformation(CL);
15841:   SIRegister_TJclImageInformation(CL);
15842:   SIRegister_TJclJpegSummaryInformation(CL);
15843: end;
15844:
15845: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15846: begin
15847:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15848:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15849:   T8087Infinity','( icProjective, icAffine )');
15850:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15851:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15852:  Function Get8087ControlWord : Word');
15853:  Function Get8087Infinity : T8087Infinity');
15854:  Function Get8087Precision : T8087Precision');
15855:  Function Get8087Rounding : T8087Rounding');
15856:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15857:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15858:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15859:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15860:  Function Set8087ControlWord( const Control : Word) : Word');
15861:  Function ClearPending8087Exceptions : T8087Exceptions');
15862:  Function GetPending8087Exceptions : T8087Exceptions');
15863:  Function GetMasked8087Exceptions : T8087Exceptions');
15864:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15865:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15866:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15867: end;
15868:
15869: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15870: begin
15871:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15872:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15873:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
       Accept:Bool;Sorted:Bool;
15874:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15875:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15876:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15877:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
15878:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15879: end;
15880:
15881: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15882: begin
15883:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15884:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15885:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15886:   type ULONG', 'Cardinal');
15887:      LPCWSTR', 'PChar');
15888:   CL.AddTypeS('LPWSTR', 'PChar');
15889:   LPSTR', 'PChar');
15890:   TBindVerb', 'ULONG');
15891:   TBindInfoF', 'ULONG');
15892:   TBindF', 'ULONG');
15893:   TBSCF', 'ULONG');
15894:   TBindStatus', 'ULONG');
15895:   TCIPStatus', 'ULONG');
15896:   TBindString', 'ULONG');
15897:   TPiFlags', 'ULONG');
15898:   TOIBdgFlags', 'ULONG');
15899:   TParseAction', 'ULONG');
15900:   TPSUAction', 'ULONG');
15901:   TQueryOption', 'ULONG');
15902:   TPUAF', 'ULONG');
15903:   TSZMFlags', 'ULONG');
15904:   TUrlZone', 'ULONG');
15905:   TUrlTemplate', 'ULONG');
15906:   TZAFlags', 'ULONG');
15907:   TUrlZoneReg', 'ULONG');
```

```
15908:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15909:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15910:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15911:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15912:  const 'CF_NULL','LongInt').SetInt( 0);
15913:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15914:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15915:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15916:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15917:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15918:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15919:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15920:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15921:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15922:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15923:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15924:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15925:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15926:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15927:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15928:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15929:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15930:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15931:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15932:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15933:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15934:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15935:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15936:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15937:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15938:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15939:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15940:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15941:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15942:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15943:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15944:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15945:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15946:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
15947:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15948:   SIRegister_IPersistMoniker(CL);
15949:   SIRegister_IBindProtocol(CL);
15950:   SIRegister_IBinding(CL);
15951:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15952:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15953:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15954:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15955:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15956:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15957:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15958:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
15959:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
15960:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
15961:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
15962:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
15963:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
15964:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
15965:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
15966:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
15967:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
15968:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
15969:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
15970:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
15971:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
15972:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
15973:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
15974:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
15975:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
15976:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
15977:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
15978:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
15979:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
15980:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
15981:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
15982:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
15983:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
15984:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
15985:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
15986:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
15987:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
15988:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
15989:  const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
15990:  const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
15991:  const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
15992:  const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
15993:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
15994:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
15995:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
15996:  const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
```

```
15997:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
15998:  const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
15999:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16000:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16001:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16002:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16003:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16004:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16005:  const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
16006:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16007:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16008:  const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
16009:  const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
16010:  const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
16011:  const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16012:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16013:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16014:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16015:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16016:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16017:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16018:  const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED + 1);
16019:  const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16020:  const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
16021:  const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16022:  const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
16023:  const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16024:  const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
16025:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16026:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16027:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16028:  const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
16029:  const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED + 1);
16030:  const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16031:  const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
16032:  const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
16033:  const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
16034:  const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16035:  // PBindInfo', '^TBindInfo // will not work');
16036:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16037:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16038:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16039:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16040:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16041:   TBindInfo', '_tagBINDINFO');
16042:   BINDINFO', '_tagBINDINFO');}
16043:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16044:    +'criptor : DWORD; bInheritHandle : BOOL; end');
16045:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
16046:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
16047:   //PRemBindInfo', '^TRemBindInfo // will not work');
16048:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16049:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16050:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16051:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16052:    +'n; dwReserved : DWORD; end');
16053:   TRemBindInfo', '_tagRemBINDINFO');
16054:   RemBINDINFO', '_tagRemBINDINFO');}
16055:   //PRemFormatEtc', '^TRemFormatEtc // will not work');
16056:   tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16057:   TRemFormatEtc', 'tagRemFORMATETC');
16058:   RemFORMATETC', 'tagRemFORMATETC');
16059:   SIRegister_IBindStatusCallback(CL);
16060:   SIRegister_IAuthenticate(CL);
16061:   SIRegister_IHttpNegotiate(CL);
16062:   SIRegister_IWindowForBindingUI(CL);
16063:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16064:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16065:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16066:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16067:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16068:  const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16069:  const 'CIP_EXE_SELF_REGISTERATION_TIMEOUT',LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
16070:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16071:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16072:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16073:   SIRegister_ICodeInstall(CL);
16074:   SIRegister_IWinInetInfo(CL);
16075:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16076:   SIRegister_IHttpSecurity(CL);
16077:   SIRegister_IWinInetHttpInfo(CL);
16078:   SIRegister_IBindHost(CL);
16079:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16080:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16081:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16082:  Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback) : HResult');
16083:  Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback) : HResult');
16084:  Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
      : IBindStatusCallback) : HResult');
```

```
16085:  Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
       IBindStatusCallback) : HResult');
16086:  Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
       p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult');
16087:  Function HlinkGoBack( unk : IUnknown) : HResult');
16088:  Function HlinkGoForward( unk : IUnknown) : HResult');
16089:  Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult');
16090: // Function HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult');
16091:   SIRegister_IInternet(CL);
16092:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16093:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16094:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16095:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16096:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16097:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16098:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16099:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16100:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16101:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16102:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16103:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16104:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16105:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16106:   //POLEStrArray', '^TOLESTRArray // will not work');
16107:   SIRegister_IInternetBindInfo(CL);
16108:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16109:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16110:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16111:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16112:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16113:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16114:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16115:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16116:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16117:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16118:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16119:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16120:   //PProtocolData', '^TProtocolData // will not work');
16121:   _tagPROTOCOLDATA','record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16122:   TProtocolData', '_tagPROTOCOLDATA');
16123:   PROTOCOLDATA', '_tagPROTOCOLDATA');
16124:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink, 'IInternetProtocolSink');
16125:   SIRegister_IInternetProtocolRoot(CL);
16126:   SIRegister_IInternetProtocol(CL);
16127:   SIRegister_IInternetProtocolSink(CL);
16128:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16129:   SIRegister_IInternetSession(CL);
16130:   SIRegister_IInternetThreadSwitch(CL);
16131:   SIRegister_IInternetPriority(CL);
16132:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16133:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16134:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16135:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16136:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16137:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16138:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16139:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16140:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16141:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16142:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16143:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16144:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16145:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16146:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16147:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16148:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16149:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16150:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16151:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16152:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16153:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16154:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16155:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16156:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16157:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16158:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16159:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16160:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16161:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16162:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16163:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16164:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16165:   SIRegister_IInternetProtocolInfo(CL);
16166:   IOInet', 'IInternet');
16167:   IOInetBindInfo', 'IInternetBindInfo');
16168:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16169:   IOInetProtocol', 'IInternetProtocol');
16170:   IOInetProtocolSink', 'IInternetProtocolSink');
16171:   IOInetProtocolInfo', 'IInternetProtocolInfo');
```

```
16172:   IOInetSession', 'IInternetSession');
16173:   IOInetPriority', 'IInternetPriority');
16174:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16175:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16176:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16177:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16178:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16179:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
       pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16180:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
       IInternetSes;dwReserved:DWORD):HResult;
16181:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16182:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16183:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16184:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16185:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16186:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
       TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16187:  Function OInetGetSession(dwSessionMode:DWORD; var
       pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16188:   //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16189:  //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16190: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16191: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16192:  //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16193:  //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16194:  //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16195:  const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16196:   SIRegister_IInternetSecurityMgrSite(CL);
16197:  const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16198:  const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16199:  const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16200:  const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16201:  const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16202:  const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16203:  const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16204:  const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16205:  const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16206:  const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16207:  const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16208:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16209:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16210:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16211:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16212:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16213:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16214:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16215:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16216:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16217:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16218:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16219:  const 'PUAF_NOUIFLOCKED','LongWord').SetUInt( $00100000);
16220:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16221:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16222:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16223:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16224:   SIRegister_IInternetSecurityManager(CL);
16225:   SIRegister_IInternetHostSecurityManager(CL);
16226:   SIRegister_IInternetSecurityManagerEx(CL);
16227:  const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16228:  const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16229:  const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16230:  const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16231:  const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16232:  const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16233:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16234:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16235:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16236:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16237:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16238:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16239:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16240:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16241:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16242:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16243:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16244:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16245:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16246:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16247:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16248:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16249:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16250:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16251:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
```

```
16252:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16253:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16254:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16255:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16256:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16257:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16258:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16259:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16260:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16261:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16262:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16263:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16264:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16265:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16266:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16267:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16268:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16269:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16270:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16271:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16272:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16273:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16274:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16275:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16276:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16277:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16278:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16279:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16280:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16281:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16282:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16283:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16284:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16285:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16286:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16287:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16288:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16289:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16290:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16291:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16292:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16293:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16294:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16295:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16296:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16297:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16298:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16299:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16300:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16301:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16302:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16303:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16304:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16305:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16306:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16307:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16308:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16309:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16310:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16311:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16312:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16313:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16314:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16315:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16316:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16317:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16318:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16319:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16320:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16321:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16322:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16323:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16324:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16325:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16326:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16327:  Function GetUrlPolicyPermissions( dw : DWORD) : DWORD');
16328:  Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD');
16329:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16330:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16331:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16332:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16333:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16334:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16335:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16336:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16337:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16338:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16339:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16340:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
```

```
16341:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16342:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16343:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16344:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16345:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16346:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16347:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16348:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16349:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16350:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16351:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16352:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16353:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16354:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16355:   //PZoneAttributes', '^TZoneAttributes // will not work');
16356:  _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');
16357:   { _ZONEATTRIBUTES = packed record
16358:     cbSize: ULONG;
16359:     szDisplayName: array [0..260 - 1] of WideChar;
16360:     szDescription: array [0..200 - 1] of WideChar;
16361:     szIconPath: array [0..260 - 1] of WideChar;
16362:     dwTemplateMinLevel: DWORD;
16363:     dwTemplateRecommended: DWORD;
16364:     dwTemplateCurrentLevel: DWORD;
16365:     dwFlags: DWORD;
16366:   end; }
16367:   TZoneAttributes', '_ZONEATTRIBUTES');
16368:   ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16369:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16370:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16371:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16372:   SIRegister_IInternetZoneManager(CL);
16373:   SIRegister_IInternetZoneManagerEx(CL);
16374:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16375:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16376:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16377:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16378:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16379:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16380:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16381:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
16382:   //PCodeBaseHold', '^TCodeBaseHold // will not work');
16383:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16384:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16385:   TCodeBaseHold', '_tagCODEBASEHOLD');
16386:   CODEBASEHOLD', '_tagCODEBASEHOLD');
16387:   //PSoftDistInfo', '^TSoftDistInfo // will not work');
16388:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
16389:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
16390:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
16391:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
16392:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16393:   TSoftDistInfo', '_tagSOFTDISTINFO');
16394:   SOFTDISTINFO', '_tagSOFTDISTINFO');
16395:   SIRegister_ISoftDistExt(CL);
16396:  Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16397:  Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
       dwAdvertisedVersionLS : DWORD) : HResult');
16398:   SIRegister_IDataFilter(CL);
16399:  // PProtocolFilterData', '^TProtocolFilterData // will not work');
16400:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
16401:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16402:    +'terFlags : DWORD; end');
16403:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16404:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16405:   //PDataInfo', '^TDataInfo // will not work');
16406:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
16407:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16408:   TDataInfo', '_tagDATAINFO');
16409:   DATAINFO', '_tagDATAINFO');
16410:   SIRegister_IEncodingFilterFactory(CL);
16411:  Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16412:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16413:  //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16414:   //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16415:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16416:    +rlName : LPSTR; StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16417:   THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16418:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16419:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16420: end;
16421:
16422: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16423: begin
16424:  Procedure reformatMemo( const m : TCustomMemo)');
16425:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16426:  Procedure MoveToTop( memo : TMemo)');
```

```
16427:  Procedure ScrollToTop( memo : TMemo)');
16428:  Function LineNumberClicked( memo : TMemo) : integer');
16429:  Function MemoClickedLine( memo : TMemo) : integer');
16430:  Function ClickedMemoLine( memo : TMemo) : integer');
16431:  Function MemoLineClicked( memo : TMemo) : integer');
16432:  Function LinePositionClicked( Memo : TMemo) : integer');
16433:  Function ClickedMemoPosition( memo : TMemo) : integer');
16434:  Function MemoPositionClicked( memo : TMemo) : integer');
16435:  Procedure AdjustGridSize( grid : TDrawGrid)');
16436:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16437:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16438:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16439:  Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');
16440:  Procedure sortstrDown( var s : string)');
16441:  Procedure sortstrUp( var s : string)');
16442:  Procedure rotatestrleft( var s : string)');
16443:  Function dffstrtofloatdef( s : string; default : extended) : extended');
16444:  Function deblank( s : string) : string');
16445:  Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16446:  Procedure FreeAndClearListBox( C : TListBox);');
16447:  Procedure FreeAndClearMemo( C : TMemo);');
16448:  Procedure FreeAndClearStringList( C : TStringList);');
16449:  Function dffgetfilesize( f : TSearchrec) : int64');
16450: end;
16451:
16452: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16453: begin
16454:   CL.AddTypeS('intset', 'set of byte');
16455:   TPoint64', 'record x : int64; y : int64; end');
16456:  Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16457:  Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16458:  Function GeneratePentagon( n : integer) : integer');
16459:  Function IsPentagon( p : integer) : boolean');
16460:  Function isSquare( const N : int64) : boolean');
16461:  Function isCube( const N : int64) : boolean');
16462:  Function isPalindrome( const n : int64) : boolean;');
16463:  Function isPalindrome1( const n : int64; var len : integer) : boolean;');
16464:  Function GetEulerPhi( n : int64) : int64');
16465:  Function dffIntPower( a, b : int64) : int64;');
16466:  Function IntPower1( a : extended; b : int64) : extended;');
16467:  Function gcd2( a, b : int64) : int64');
16468:  Function GCDMany( A : array of integer) : integer');
16469:  Function LCMMany( A : array of integer) : integer');
16470:  Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16471:  Function dffFactorial( n : int64) : int64');
16472:  Function digitcount( n : int64) : integer');
16473:  Function nextpermute( var a : array of integer) : boolean');
16474:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16475:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16476:  Function InttoBinaryStr( nn : integer) : string');
16477:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16478:  Function AngleToStr( angle : extended) : string');
16479:  Function deg2rad( deg : extended) : extended');
16480:  Function rad2deg( rad : extended) : extended');
16481:  Function GetLongToMercProjection( const long : extended) : extended');
16482:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16483:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16484:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16485:   SIRegister_TPrimes(CL);
16486:   //RIRegister_TPrimes(CL);
16487:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16488:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16489:  Function Random64( const N : Int64) : Int64;');
16490:  Procedure Randomize64');
16491:  Function Random641 : extended;');
16492:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16493: end;
16494:
16495: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16496: begin
16497:   TrealPoint', 'record x : extended; y : extended; end');
16498:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16499:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16500:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16501:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16502:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16503:  Function realpoint( x, y : extended) : TRealPoint');
16504:  Function dist( const p1, p2 : TrealPoint) : extended');
16505:  Function intdist( const p1, p2 : TPoint) : integer');
16506:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16507:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16508:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16509:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16510:  Function GetTheta( const L : TLine) : extended;');
16511:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16512:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16513:  Procedure Extendline( var L : TLine; dist : integer);');
16514:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16515:  Function Linesintersect( line1, line2 : TLine) : boolean');
```

```
16516:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16517:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16518:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16519:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16520:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16521:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16522:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16523:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16524:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16525:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16526:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16527:  Function DegtoRad( d : extended) : extended');
16528:  Function RadtoDeg( r : extended) : extended');
16529:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16530:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16531:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16532:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16533:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
16534:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16535:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16536:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16537:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16538: end;
16539:
16540:
16541: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16542: begin
16543:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16544:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16545:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16546:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16547:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16548:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16549:    +'arth : extended; Phase : extended; end');
16550:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16551:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16552:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16553:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
16554:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16555:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16556:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16557:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16558:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16559:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended; UncorrectedEarthDistance : extended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended;
       ApparentRaDecl:TRPoint; end');
16560:   SIRegister_TAstronomy(CL);
16561:  Function AngleToStr( angle : extended) : string');
16562:  Function StrToAngle( s : string; var angle : extended) : boolean');
16563:  Function HoursToStr24( t : extended) : string');
16564:  Function RPoint( x, y : extended) : TRPoint');
16565:  Function getStimename( t : TDTType) : string');
16566: end;
16567:
16568: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16569: begin
16570:   TCardValue', 'Integer');
16571:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16572:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16573:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16574:   SIRegister_TCard(CL);
16575:   SIRegister_TDeck(CL);
16576: end;
16577:
16578: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16579: begin
16580:   tMethodCall', 'Procedure');
16581:   tVerboseCall', 'Procedure ( s : string)');
16582:  // PTEdge', '^TEdge // will not work');
16583:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16584:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16585:   SIRegister_TNode(CL);
16586:   SIRegister_TGraphList(CL);
16587: end;
16588:
16589: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16590: begin
16591:   ParserFloat', 'extended');
16592:   //PParserFloat', '^ParserFloat // will not work');
16593:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16594:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16595:   //POperation', '^TOperation // will not work');
16596:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16597:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
```

```
16598:    +'; Token : TDFFToken; end');
16599:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16600:   (CL.FindClass('TOBJECT'),'EMathParserError');
16601:   CL.FindClass('TOBJECT'),'ESyntaxError');
16602:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16603:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16604:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16605:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16606:   (CL.FindClass('TOBJECT'),'EBadName');
16607:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16608:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16609:   SIRegister_TCustomParser(CL);
16610:   SIRegister_TExParser(CL);
16611: end;
16612:
16613:   function isService: boolean;
16614:   begin
16615:     result:= NOT(Application is TApplication);
16616:     {result:= Application is TServiceApplication;}
16617:   end;
16618:   function isApplication: boolean;
16619:   begin
16620:     result:= Application is TApplication;
16621:   end;
16622:   //SM_REMOTESESSION = $1000
16623:   function isTerminalSession: boolean;
16624:   begin
16625:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16626:   end;
16627:
16628: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16629: begin
16630:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16631:    +'String; margin_bottom : String; margin_left : String; margin_right : Strin'
16632:    +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16633:  Function cyURLEncode( const S : string) : string');
16634:  Function MakeResourceURL(const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16635:  Function MakeResourceURL1(const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;;
16636:  Function cyColorToHtml( aColor : TColor) : String');
16637:  Function HtmlToColor( aHtmlColor : String) : TColor');
16638:  //Function GetStreamEncoding( aStream : TStream) : TEncoding');
16639:  // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) : Boolean');
16640:  Function AddHtmlUnicodePrefix( aHtml : String) : String');
16641:  Function RemoveHtmlUnicodePrefix( aHtml : String) : String');
16642:  Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
16643:  Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16644:  CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16645:  CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16646:  CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16647:  CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16648:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16649:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
16650: end;
16651:
16652:
16653: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16654: begin
16655:  CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16656:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16657:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16658:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16659:    +'inationsRepeat, CombinationsRepeatDown )');
16660:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16661:   SIRegister_TComboSet(CL);
16662: end;
16663:
16664:  procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16665: begin
16666:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
16667:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )');
16668:   TProcOnReceiveCommand', 'Procedure (Sender: TObject; aCommand: Word; userParam : Integer)');
16669:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16670:    +'mmHandle : THandle; aString : String; userParam : Integer)');
16671:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16672:    +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer)');
16673:   SIRegister_TcyBaseComm(CL);
16674:  CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16675:  CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16676:  Function ValidateFileMappingName( aName : String) : String');
16677:  procedure makeCaption(leftSide, Rightside:string; form:TForm);
16678: end;
16679:
16680: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16681: begin
16682:   CL.AddTypeS('DERString', 'String');
16683:   CL.AddTypeS('DERChar', 'Char');
16684:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16685:    +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16686:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
```

```
16687:   CL.AddTypeS('DERNString', 'String');
16688:  const DERDecimalSeparator','String').SetString( '.');
16689:  const DERDefaultChars','String')('+°@/%-
_.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16690:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz');
16691:  CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char) : Boolean');
16692:  Function isValidWebMailChar( aChar : Char) : Boolean');
16693:  Function isValidwebSite( aStr : String) : Boolean');
16694:  Function isValidWebMail( aStr : String) : Boolean');
16695:  Function ValidateDate( aDERStr : DERString; var RsltFormat : String) : Boolean');
16696:  Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16697:  Function IsDERChar( aChar : Char) : Boolean');
16698:  Function IsDERDefaultChar( aChar : Char) : Boolean');
16699:  Function IsDERMoneyChar( aChar : Char) : Boolean');
16700:  Function IsDERExceptionCar( aChar : Char) : Boolean');
16701:  Function IsDERSymbols( aDERString : String) : Boolean');
16702:  Function StringToDERCharSet( aStr : String) : DERString');
16703:  Function IsDERNDefaultChar( aChar : Char) : Boolean');
16704:  Function IsDERNChar( aChar : Char) : Boolean');
16705:  Function DERToDERNCharset( aDERStr : DERString) : DERNString');
16706:  Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean) : String');
16707:  Function DERExtractWebMail( aDERStr : DERString) : String');
16708:  Function DERExtractPhoneNr( aDERStr : DERString) : String');
16709:  Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
       SmartWebsiteRecognition:Boolean):TElementsType;
16710:  Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
       RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
       SmartWebsiteRecognition:Boolean):TElementsType;
16711:  Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
       aElementsType : TElementsType) : String;');
16712:  Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
       Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType);');
16713: end;
16714:
16715: procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16716: begin
16717:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16718:  Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16719:  Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16720:  Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16721:  Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16722:  Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16723:  Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
16724:  Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16725:  Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
       RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16726:  Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
       PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
       PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16727:  Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
       SingleDestinationColor, RefreshBmp : Boolean)');
16728:  Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
16729:  Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16730:  Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16731:  Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16732:  Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean);');
16733:  Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16734:  Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16735:  Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');
16736: end;
16737:
16738: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16739: begin
16740:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16741:   TPCMChannel', '( cMono, cStereo )');
16742:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16743:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16744:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
16745:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16746:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16747:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16748:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16749:    +'it48000Hz, Stereo16bit48000Hz )');
16750:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16751:    +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end');
16752:    tWaveFormatEx', 'PWaveFormatEx');
16753:    HMMIO', 'Integer');
16754:   TWaveDeviceFormats', 'set of TPCMFormat');
16755:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16756:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16757:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16758:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16759:   TWaveOutOptions', 'set of TWaveOutOption');
16760:   TStreamOwnership2', '( soReference, soOwned )');
16761:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16762:  // PRawWave', '^TRawWave // will not work');
16763:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16764:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioError');
16765:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError');
```

```
16766:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation');
16767:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16768:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16769:    +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16770:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16771:    +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16772:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16773:    +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16774:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16775:    +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16776:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16777:   TWaveAudioFilterEvent', 'Procedure (Sender : TObject; const Buffer:TObject; BufferSize:DWORD);
16778:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16779:  CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16780:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16781:  GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16782:  CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16783:  OpenStreamWaveAudio( Stream : TStream) : HMMIO');
16784:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16785:  GetAudioFormat( FormatTag : Word) : String');
16786:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) : String');
16787:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD');
16788:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD');
16789:  GetWaveAudioPeakLevel(const Data: TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16790:  InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16791:  SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16792:  ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16793:  MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
       TObject; BufferSize : DWORD) : Boolean');
16794:  ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
       dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16795:  SetPCMAudioFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
       TPCMSamplesPerSec; BitsPerSample : TPCMBitsPerSample)');
16796:  Procedure SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat)');
16797:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat');
16798:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD) : DWORD');
16799:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) : String');
16800:  WaitForSyncObject( SyncObject : THandle; Timeout : DWORD) : DWORD');
16801:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT');
16802: end;
16803:
16804: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);
16805: begin
16806:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16807:  CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16808:  'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
16809:  'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
16810:  'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
16811:  'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
16812:  'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
16813:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16814:   CL.AddTypeS('TPipeType', '( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg )');
16815:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16816:   SIRegister_TNamedPipe(CL);
16817:   SIRegister_TServerPipe(CL);
16818:   SIRegister_TClientPipe(CL);
16819:  CL.AddDelphiFunction('Function CalculateTimeout( aBasis : DWORD) : DWORD');
16820:  Function HasOverlappedIoCompleted( const ov : OVERLAPPED) : Boolean');
16821:  Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean):
       TOverlappedResult;
16822:  Function GetStreamAsText( stm : TStream) : string');
16823:  Procedure SetStreamAsText( const aTxt : string; stm : TStream)');
16824: end;
16825:
16826: procedure SIRegister_DPUtils(CL: TPSPascalCompiler);
16827: begin
16828:  // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16829:   SIRegister_TThumbData(CL);
16830:  'PIC_BMP','LongInt').SetInt( 0);
16831:  'PIC_JPG','LongInt').SetInt( 1);
16832:  'THUMB_WIDTH','LongInt').SetInt( 60);
16833:  'THUMB_HEIGHT','LongInt').SetInt( 60);
16834:  Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean');
16835:  Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)');
16836:  Function ReadBitmap( Filehandle : Integer; Width, Height : Integer) : TBitmap');
16837:  Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap)');
16838:  Function OpenPicture( fn : string; var tp : Integer) : Integer');
16839:  Function ConvertPicture( pi : Integer; tp : Integer) : TBitmap');
16840:  Function LoadPicture( fn : string; var w, h : Integer) : TBitmap');
16841:  Function TurnBitmap( bmp : TBitmap; ang : Integer) : TBitmap');
16842:  Function RotateBitmap( Bitmap : TBitmap; Direction : Integer) : TBitmap');
16843:  Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect');
16844:  Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer) : TBitmap');
16845:  Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer)');
16846:  Procedure FindFiles( path, mask : string; items : TStringList)');
16847:  Function LetFileName( s : string) : string');
16848:  Function LetParentPath( path : string) : string');
16849:  Function AddBackSlash( path : string) : string');
16850:  Function CutBackSlash( path : string) : string');
```

```
16851: end;
16852:
16853: procedure SIRegister_CommonTools(CL: TPSPascalCompiler);
16854: begin
16855:  //'BYTES','LongInt').SetInt( 1);
16856:  'KBYTES','LongInt').SetInt( 1024);
16857:  'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ));
16858:  'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ));
16859:  'DBG_GONE','LongWord').SetUInt( $99AC1D99);
16860:  'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D}');
16861:   SIRegister_MakeComServerMethodsPublic(CL);
16862:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16863:  Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) : Boolean');
16864:  Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean)');
16865:  Function TBGetTempFolder : string');
16866:  Function TBGetTempFile : string');
16867:  Function TBGetModuleFilename : string');
16868:  Function FormatModuleVersionInfo( const aFilename : string) : string');
16869:  Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD) : string');
16870:  Function TBGetFileSize( aFile : string; aMultipleOf : Integer) : Integer');
16871:  Function FormatAttribString( aAttr : Integer) : string');
16872:  Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo) : string');
16873:  Function ShellRecycle( aWnd : HWND; aFileOrFolder : string) : Boolean');
16874:  Function IsDebuggerPresent : BOOL');
16875:  Function TBNotImplemented : HRESULT');
16876: end;
16877:
16878: procedure SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
16879: begin
16880:   //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16881:   //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16882:   CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16883:   CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean;');
16884:   //TDrivesProperty = array['A'..'Z'] of boolean;
16885:  Function TBSetSystemTime( DateTime : TDateTime; DOW : word) : boolean');
16886:  Function IsElevated : Boolean');
16887:  Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
16888:   CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined )');
16889:  Function TrimNetResource( UNC : string) : string');
16890:  Procedure GetFreeDrives( var FreeDrives : TDrivesProperty)');
16891:  Procedure GetMappedDrives( var MappedDrives : TDrivesProperty)');
16892:  Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
       UserUsage : TPasswordUsage; User : string; Comment : string) : boolean');
16893:  Function UnmapDrive( Drive : char; Force : boolean) : boolean');
16894:  Function TBIsWindowsVista : Boolean');
16895:  Procedure SetVistaFonts( const AForm : TForm)');
16896:  Procedure SetVistaContentFonts( const AFont : TFont)');
16897:  Function GetProductType( var sType : String) : Boolean');
16898:  Function lstrcmp( lpString1, lpString2 : PChar) : Integer');
16899:  Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16900:  Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar');
16901:  Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
16902:  Function lstrcat( lpString1, lpString2 : PChar) : PChar');
16903:  Function lstrlen( lpString : PChar) : Integer');
16904:  Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16905:  Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD) : BOOL');
16906: end;
16907:
16908: procedure SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
16909: begin
16910:  'cLineTerminator','Char').SetString( #10);
16911:  'cLineTerminators','String').SetString( #13#10);
16912:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
16913:   SIRegister_TFixedCriticalSection(CL);
16914:   SIRegister_TMultiReadSingleWrite(CL);
16915:  CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16916:  Function GetDecimalSeparator : Char');
16917:  TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16918:  Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
       recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16919:   CL.AddTypeS('NativeInt', 'Integer');
16920:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16921:   CL.AddTypeS('NativeUInt', 'Cardinal');
16922:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16923:   //CL.AddTypeS('TBytes', 'array of Byte');
16924:   CL.AddTypeS('RawByteString', 'UnicodeString');
16925:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16926:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16927:   SIRegister_TPath(CL);
16928:   SIRegister_TFile(CL);
16929:   SIRegister_TdwsThread(CL);
16930:  Function GetSystemMilliseconds : Int64');
16931:  Function UTCDateTime : TDateTime');
16932:  Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString');
16933:  Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer');
16934:  Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer');
16935:  Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer');
```

```
16936:  Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer;');
16937:  Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer;');
16938:  Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString');
16939:  Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString');
16940:  Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer');
16941:  Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean');
16942:  Function InterlockedIncrement( var val : Integer) : Integer');
16943:  Function InterlockedDecrement( var val : Integer) : Integer');
16944:  Procedure FastInterlockedIncrement( var val : Integer)');
16945:  Procedure FastInterlockedDecrement( var val : Integer)');
16946:  Function InterlockedExchangePointer( var target : ___Pointer; val : ___Pointer) : ___Pointer');
16947:  Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
16948:  Procedure dwsOutputDebugString( const msg : UnicodeString)');
16949:  Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16950:  Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16951:  Procedure VarCopy( out dest : Variant; const src : Variant)');
16952:  Function VarToUnicodeStr( const v : Variant) : UnicodeString');
16953:  Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString');
16954:  Function LoadTextFromStream( aStream : TStream) : UnicodeString');
16955:  Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString');
16956:  Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
16957:  Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle');
16958:  Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle');
16959:  Procedure CloseFileHandle( hFile : THandle)');
16960:  Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean');
16961:  Function FileMove( const existing, new : UnicodeString) : Boolean');
16962:  Function dwsFileDelete( const fileName : String) : Boolean');
16963:  Function FileRename( const oldName, newName : String) : Boolean');
16964:  Function dwsFileSize( const name : String) : Int64');
16965:  Function dwsFileDateTime( const name : String) : TDateTime');
16966:  Function DirectSet8087CW( newValue : Word) : Word');
16967:  Function DirectSetMXCSR( newValue : Word) : Word');
16968:  Function TtoObject( const T: byte) : TObject');
16969:  Function TtoPointer( const T: byte) : ___Pointer');
16970:  Procedure GetMemForT(var T: byte; Size : integer)');
16971:  Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');
16972: end;
16973:
16974: procedure SIRegister_AdSocket(CL: TPSPascalCompiler);
16975: begin
16976:  'IPStrSize','LongInt').SetInt( 15);
16977:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
16978:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
16979:  'ADWSBASE','LongInt').SetInt( 9000);
16980:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
16981:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
16982:   SIRegister_EApdSocketException(CL);
16983:   TWsMode', '( wsClient, wsServer )');
16984:   TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket');
16985:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrCode : Integer)');
16986:   SIRegister_TApdSocket(CL);
16987: end;
16988:
16989: procedure SIRegister_AdPort(CL: TPSPascalCompiler);
16990: begin
16991:   SIRegister_TApdCustomComPort(CL);
16992:   SIRegister_TApdComPort(CL);
16993:  Function ComName( const ComNumber : Word) : ShortString');
16994:  Function SearchComPort( const C : TComponent) : TApdCustomComPort');
16995: end;
16996:
16997: procedure SIRegister_PathFunc(CL: TPSPascalCompiler);
16998: begin
16999:  Function inAddBackslash( const S : String) : String');
17000:  Function PathChangeExt( const Filename, Extension : String) : String');
17001:  Function PathCharCompare( const S1, S2 : PChar) : Boolean');
17002:  Function PathCharIsSlash( const C : Char) : Boolean');
17003:  Function PathCharIsTrailByte( const S : String; const Index : Integer) : Boolean');
17004:  Function PathCharLength( const S : String; const Index : Integer) : Integer');
17005:  Function inPathCombine( const Dir, Filename : String) : String');
17006:  Function PathCompare( const S1, S2 : String) : Integer');
17007:  Function PathDrivePartLength( const Filename : String) : Integer');
17008:  Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;
17009:  Function inPathExpand( const Filename : String) : String');
17010:  Function PathExtensionPos( const Filename : String) : Integer');
17011:  Function PathExtractDir( const Filename : String) : String');
17012:  Function PathExtractDrive( const Filename : String) : String');
17013:  Function PathExtractExt( const Filename : String) : String');
17014:  Function PathExtractName( const Filename : String) : String');
17015:  Function PathExtractPath( const Filename : String) : String');
17016:  Function PathIsRooted( const Filename : String) : Boolean');
17017:  Function PathLastChar( const S : String) : PChar');
17018:  Function PathLastDelimiter( const Delimiters, S : string) : Integer');
17019:  Function PathLowercase( const S : String) : String');
17020:  Function PathNormalizeSlashes( const S : String) : String');
17021:  Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
17022:  Function PathPos( Ch : Char; const S : String) : Integer');
17023:  Function PathStartsWith( const S, AStartsWith : String) : Boolean');
17024:  Function PathStrNextChar( const S : PChar) : PChar');
```

```
17025:  Function PathStrPrevChar( const Start, Current : PChar) : PChar');
17026:  Function PathStrScan( const S : PChar; const C : Char) : PChar');
17027:  Function inRemoveBackslash( const S : String) : String');
17028:  Function RemoveBackslashUnlessRoot( const S : String) : String');
17029:  Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean)');
17030: end;
17031:
17032:
17033: procedure SIRegister_CmnFunc2(CL: TPSPascalCompiler);
17034: begin
17035:  NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17036:  NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17037:  NEWREGSTR_PATH_SPECIAL_FOLDERS','String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17038:  NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17039:  NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName');
17040:  NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString');
17041:  KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
17042:   //CL.AddTypeS('PLeadByteSet', '^TLeadByteSet // will not work');
17043:   CL.AddTypeS('TLeadByteSet', 'set of Char');
17044:   SIRegister_TOneShotTimer(CL);
17045:   CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17046:  'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17047:  Function NewFileExists( const Name : String) : Boolean');
17048:  Function inDirExists( const Name : String) : Boolean');
17049:  Function FileOrDirExists( const Name : String) : Boolean');
17050:  Function IsDirectoryAndNotReparsePoint( const Name : String) : Boolean');
17051:  Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String');
17052:  Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17053:  Function GetIniBool( const Section,Key:String; const Default:Boolean;const Filename:String): Boolean');
17054:  Function IniKeyExists( const Section, Key, Filename : String) : Boolean');
17055:  Function IsIniSectionEmpty( const Section, Filename : String) : Boolean');
17056:  Function SetIniString( const Section, Key, Value, Filename : String) : Boolean');
17057:  Function SetIniInt(const Section,Key:String;const Value: Longint;const Filename: String):Boolean');
17058:  Function SetIniBool(const Section,Key:String;const Value: Boolean;const Filename: String):Boolean');
17059:  Procedure DeleteIniEntry( const Section, Key, Filename : String)');
17060:  Procedure DeleteIniSection( const Section, Filename : String)');
17061:  Function GetEnv( const EnvVar : String) : String');
17062:  Function GetCmdTail : String');
17063:  Function GetCmdTailEx( StartIndex : Integer) : String');
17064:  Function NewParamCount : Integer');
17065:  Function NewParamStr( Index : Integer) : string');
17066:  Function AddQuotes( const S : String) : String');
17067:  Function RemoveQuotes( const S : String) : String');
17068:  Function inGetShortName( const LongName : String) : String');
17069:  Function inGetWinDir : String');
17070:  Function inGetSystemDir : String');
17071:  Function GetSysWow64Dir : String');
17072:  Function GetSysNativeDir( const IsWin64 : Boolean) : String');
17073:  Function inGetTempDir : String');
17074:  Function StringChange( var S : String; const FromStr, ToStr : String) : Integer');
17075:  Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17076:  Function AdjustLength( var S : String; const Res : Cardinal) : Boolean');
17077:  Function UsingWinNT : Boolean');
17078:  Function ConvertConstPercentStr( var S : String) : Boolean');
17079:  Function ConvertPercentStr( var S : String) : Boolean');
17080:  Function ConstPos( const Ch : Char; const S : String) : Integer');
17081:  Function SkipPastConst( const S : String; const Start : Integer) : Integer');
17082:  Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String) : Boolean');
17083:  Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String):Boolean;
17084:  Function RegValueExists( H : HKEY; Name : PChar) : Boolean');
17085:  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;
17086:  Function RegOpenKeyExView(const
       RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17087:  Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar) : Longint;
17088:  Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17089:  Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyName:PChar):Longint;
17090:  Function GetShellFolderPath( const FolderID : Integer) : String');
17091:  Function IsAdminLoggedOn : Boolean');
17092:  Function IsPowerUserLoggedOn : Boolean');
17093:  Function IsMultiByteString( const S : AnsiString) : Boolean');
17094:  Function FontExists( const FaceName : String) : Boolean');
17095:  //Procedure FreeAndNil( var Obj)');
17096:  Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT) : HMODULE');
17097:  Function GetUILanguage : LANGID');
17098:  Function RemoveAccelChar( const S : String) : String');
17099:  Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17100:  Function AddPeriod( const S : String) : String');
17101:  Function GetExceptMessage : String');
17102:  Function GetPreferredUIFont : String');
17103:  Function IsWildcard( const Pattern : String) : Boolean');
17104:  Function WildcardMatch( const Text, Pattern : PChar) : Boolean');
17105:  Function IntMax( const A, B : Integer) : Integer');
17106:  Function Win32ErrorString( ErrorCode : Integer) : String');
17107:  Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet)');
17108:  Function inCompareMem( P1, P2 : TObject; Length : Integer) : Boolean');
17109:  Function DeleteDirTree( const Dir : String) : Boolean');
17110:  Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
```

```
17111:  Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT)');
17112:  // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17113:  Function inCharInSet( C : Char; const CharSet : TSysCharSet) : Boolean');
17114:  Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String) : Boolean');
17115:  Function ShutdownBlockReasonDestroy( Wnd : HWND) : Boolean');
17116:  Function TryStrToBoolean( const S : String; var BoolResult : Boolean) : Boolean');
17117:  Procedure WaitMessageWithTimeout( const Milliseconds : DWORD)');
17118:  Function MoveFileReplace(const ExistingFileName, NewFileName : String) : Boolean');
17119:  Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND)');
17120: end;
17121:
17122: procedure SIRegister_CmnFunc(CL: TPSPascalCompiler);
17123: begin
17124:   SIRegister_TWindowDisabler(CL);
17125:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17126:   TMsgBoxCallbackFunc','procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17127:  Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox)');
17128:  Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17129:  Function AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Integer');
17130:  Function MsgBoxP( const Text,Caption: PChar; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17131:  Function inMsgBox(const Text,Caption:String; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17132:  Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
       Typ:TMsgBoxType;const Buttons:Cardinal):Integer');
17133:  Procedure ReactivateTopWindow');
17134:  Procedure SetMessageBoxCaption( const Typ : TMsgBoxType; const NewCaption : PChar)');
17135:  Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean)');
17136:  Procedure SetMessageBoxCallbackFunc(const AFunc : TMsgBoxCallbackFunc; const AParam : LongInt)');
17137: end;
17138:
17139: procedure SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17140: begin
17141:   SIRegister_TImageGrabber(CL);
17142:   SIRegister_TCaptureDrivers(CL);
17143:   SIRegister_TCaptureDriver(CL);
17144: end;
17145:
17146: procedure SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17147: begin
17148:  Function GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:String;const
       Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean');
17149:  Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
       Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean');
17150: end;
17151:
17152: procedure SIRegister_RedirFunc(CL: TPSPascalCompiler);
17153: begin
17154:   CL.AddTypeS('TPreviousFsRedirectionState', record DidDisable : Boolean; OldValue : ___Pointer; end');
17155:  CL.AddDelphiFunction('Function AreFsRedirectionFunctionsAvailable : Boolean');
17156:  Function DisableFsRedirectionIf(const Disable:Boolean;var PreviousState:TPreviousFsRedirectionState):Boo;
17157:  Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState)');
17158:  Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17159:  Function CreateProcessRedir(const DisableFsRedir:Boolean;const lpApplicationName:PChar;const
       lpCommandLine:PChar; const lpProcessAttributes, lpThreadAttributes:PSecurityAttributes;const
       bInheritHandles:BOOL; const dwCreationFlags:DWORD;const lpEnvironment:Pointer; const
       lpCurrentDirectory:PChar; const lpStartupInfo : TStartupInfo; var
       lpProcessInformation:TProcessInformation):BOOL;
17160:  Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
       FailIfExists : BOOL) : BOOL');
17161:  Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17162:  Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17163:  Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17164:  Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
       TWin32FindData) : THandle');
17165:  Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String) : DWORD');
17166:  Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String) : String');
17167:  Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
       TFileVersionNumbers) : Boolean');
17168:  Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17169:  Function MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
17170:  Function MoveFileExRedir(const DisableFsRedir:Boolean;const ExistingFilename,NewFilename:String;const
       Flags:DWORD):BOOL;
17171:  Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17172:  Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17173:  Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17174:  Function SetNTFSCompressionRedir(const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17175:   SIRegister_TFileRedir(CL);
17176:   SIRegister_TTextFileReaderRedir(CL);
17177:   SIRegister_TTextFileWriterRedir(CL);
17178: end;
17179:
17180: procedure SIRegister_Int64Em(CL: TPSPascalCompiler);
17181: begin
17182:   //CL.AddTypeS('LongWord', 'Cardinal');
17183:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17184:  Function Compare64( const N1, N2 : Integer64) : Integer');
17185:  Procedure Dec64( var X : Integer64; N : LongWord)');
17186:  Procedure Dec6464( var X : Integer64; const N : Integer64)');
17187:  Function Div64( var X : Integer64; const Divisor : LongWord) : LongWord');
17188:  Function Inc64( var X : Integer64; N : LongWord) : Boolean');
```

```
17189:  Function Inc6464( var X : Integer64; const N : Integer64) : Boolean');
17190:  Function Integer64ToStr( X : Integer64) : String');
17191:  Function Mod64( const X : Integer64; const Divisor : LongWord) : LongWord');
17192:  Function Mul64( var X : Integer64; N : LongWord) : Boolean');
17193:  Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64)');
17194:  Procedure Shr64( var X : Integer64; Count : LongWord)');
17195:  Function StrToInteger64( const S : String; var X : Integer64) : Boolean');
17196: end;
17197:
17198: procedure SIRegister_InstFunc(CL: TPSPascalCompiler);
17199: begin
17200:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17201:   SIRegister_TSimpleStringList(CL);
17202:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle)');
17203:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17204:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17205:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17206:  // TMD5Digest = array[0..15] of Byte;
17207:   //     TSHA1Digest = array[0..19] of Byte;
17208:  Function CheckForMutexes( Mutexes : String) : Boolean');
17209:  Function CreateTempDir : String');
17210:  Function DecrementSharedCount( const RegView : TRegView; const Filename : String) : Boolean');
17211:  Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
       FirstRetryDelayMS, SubsequentRetryDelayMS : Integer)');
17212:  //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
       DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
       TDeleteFileProc; const Param : Pointer) : Boolean');
17213:  //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
       TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer) :
       TDetermineDefaultLanguageResult');
17214:  //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17215:  Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String) : Boolean');
17216:  Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17217:  Function GetComputerNameString : String');
17218:  Function GetFileDateTime(const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime):Bool;
17219:  Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String) : TMD5Digest');
17220:  Function GetMD5OfAnsiString( const S : AnsiString) : TMD5Digest');
17221: // Function GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest');
17222:  Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest');
17223:  Function GetSHA1OfAnsiString( const S : AnsiString) : TSHA1Digest');
17224: // Function GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest');
17225:  Function GetRegRootKeyName( const RootKey : HKEY) : String');
17226:  Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
       TotBytes:Int64):Bool;
17227:  Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
       TotalBytes: Integer64):Bool;
17228:  Function GetUserNameString : String');
17229:  Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17230:  //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
       String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
       ResultCode:Integer) : Boolean');
17231: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
       TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17232:  Procedure InternalError( const Id : String)');
17233:  Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17234:  Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean');
17235:  Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17236:  Function MakePendingFileRenameOperationsChecksum : TMD5Digest');
17237:  Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17238:  Procedure RaiseFunctionFailedError( const FunctionName : String)');
17239:  Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17240:  Procedure RefreshEnvironment');
17241:  Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17242:  Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17243:  Procedure UnregisterFont( const FontName, FontFilename : String)');
17244:  Function RestartComputer : Boolean');
17245:  Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17246:  Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String)');
17247:  Procedure Win32ErrorMsg( const FunctionName : String)');
17248:  Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17249:  Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17250:  //from Func2
17251:  //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
       IconFilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean; '
17252:   //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String');
17253:  Procedure RegisterTypeLibrary( const Filename : String)');
17254:  //Procedure UnregisterTypeLibrary( const Filename : String)');
17255:  //Function UnpinShellLink( const Filename : String) : Boolean');}
17256:  function getVersionInfoEx3: TOSVersionInfoEx;');
17257:  Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17258:  procedure InitOle;');
17259:  Function ExpandConst( const S : String) : String');
17260:  Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17261:  Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17262:  Function ExpandConstIfPrefixed( const S : String) : String');
17263:  Procedure LogWindowsVersion');
17264:  Function EvalCheck( const Expression : String) : Boolean');
17265: end;
17266:
```

```
17267: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17268: begin
17269:   CL.AddClassN(CL.FindClass('TOBJECT'),'TResourceDetails');
17270:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17271:   SIRegister_TResourceModule(CL);
17272:   SIRegister_TResourceDetails(CL);
17273:   SIRegister_TAnsiResourceDetails(CL);
17274:   SIRegister_TUnicodeResourceDetails(CL);
17275:  Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17276:  Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17277:  Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17278:  Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17279:  Function ResourceNameToInt( const s : string) : Integer');
17280:  Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer');
17281: end;
17282:
17283:
17284: procedure SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17285: begin
17286:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17287:   with CL.AddClassN(CL.FindClass('TObject'),'TSimpleComPort') do begin
17288:     RegisterMethod('Constructor Create');
17289:     RegisterMethod('Procedure Free');
17290:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17291:     RegisterMethod('Procedure WriteString( const S : String)');
17292:     RegisterMethod('Procedure ReadString( var S : String)');
17293:   end;
17294:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17295:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17296:       SimpleComPort.WriteString(AsciiChar);
17297: end;
17298:
17299:
17300: procedure SIRegister_Console(CL: TPSPascalCompiler);
17301: begin
17302:  CL.AddConstantN('White','LongInt').SetInt( 15); }
17303: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17304:  CL.AddConstantN('conBW40','LongInt').SetInt( 0);
17305:  CL.AddConstantN('conCO40','LongInt').SetInt( 1);
17306:  CL.AddConstantN('conBW80','LongInt').SetInt( 2);
17307:  CL.AddConstantN('conCO80','LongInt').SetInt( 3);
17308:  CL.AddConstantN('conMono','LongInt').SetInt( 7);
17309:  CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17310:  //CL.AddConstantN('C40','').SetString( C040);
17311:  //CL.AddConstantN('C80','').SetString( C080);
17312:  Function conReadKey : Char');
17313:  Function conKeyPressed : Boolean');
17314:  Procedure conGotoXY( X, Y : Smallint)');
17315:  Function conWhereX : Integer');
17316:  Function conWhereY : Integer');
17317:  Procedure conTextColor( Color : Byte);');
17318:  Function conTextColor1 : Byte;');
17319:  Procedure conTextBackground( Color : Byte);');
17320:  Function conTextBackground1 : Byte;');
17321:  Procedure conTextMode( Mode : Word)');
17322:  Procedure conLowVideo');
17323:  Procedure conHighVideo');
17324:  Procedure conNormVideo');
17325:  Procedure conClrScr');
17326:  Procedure conClrEol');
17327:  Procedure conInsLine');
17328:  Procedure conDelLine');
17329:  Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17330:  Function conScreenWidth : Smallint');
17331:  Function conScreenHeight : Smallint');
17332:  Function conBufferWidth : Smallint');
17333:  Function conBufferHeight : Smallint');
17334:  procedure InitScreenMode;');
17335: end;
17336:
17337: (*----------------------------------------------------------------------------*)
17338: procedure SIRegister_testutils(CL: TPSPascalCompiler);
17339: begin
17340:   SIRegister_TNoRefCountObject(CL);
17341:  Procedure FreeObjects( List : TFPList)');
17342:  Procedure GetMethodList( AObject : TObject; AList : TStrings);');
17343:  Procedure GetMethodList1( AClass : TClass; AList : TStrings);');
17344: end;
17345:
17346: procedure SIRegister_ToolsUnit(CL: TPSPascalCompiler);
17347: begin
17348:  'MaxDataSet','LongInt').SetInt( 35);
17349:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17350:   SIRegister_TDBConnector(CL);
17351:   SIRegister_TDBBasicsTestSetup(CL);
17352:   SIRegister_TTestDataLink(CL);
17353:  'testValuesCount','LongInt').SetInt( 25);
17354:  Procedure InitialiseDBConnector');
17355:  Procedure FreeDBConnector');
```

```
17356:  Function DateTimeToTimeString( d : tdatetime) : string');
17357:  Function TimeStringToDateTime( d : String) : TDateTime');
17358: end;
17359:
17360: procedure SIRegister_fpcunit(CL: TPSPascalCompiler);
17361: begin
17362:   SIRegister_EAssertionFailedError(CL);
17363:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17364:   CL.AddTypeS('TRunMethod', 'Procedure');
17365:   CL.AddClassN(CL.FindClass('TOBJECT'),'TTestResult');
17366:   SIRegister_TTest(CL);
17367:   SIRegister_TAssert(CL);
17368:   SIRegister_TTestFailure(CL);
17369:   SIRegister_ITestListener(CL);
17370:   SIRegister_TTestCase(CL);
17371:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17372:   SIRegister_TTestSuite(CL);
17373:   SIRegister_TTestResult(CL);
17374:  Function ComparisonMsg( const aExpected : string; const aActual : string) : string');
17375: end;
17376:
17377: procedure SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
17378: begin
17379:   TOBJECT'),'ETCPBuffer');
17380:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Intege'
17381:    +r; Head : Integer; Used : Integer; end');
17382:  'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
17383:  'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
17384:  'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBIT * 4);
17385:  'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4);
17386:  Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int;const TCPBufSize:Int;
17387:  Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer)');
17388:  Procedure TCPBufferPack( var TCPBuf : TTCPBuffer)');
17389:  Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer)');
17390:  Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer)');
17391:  Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer)');
17392:  Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer) : Pointer');
17393:  Procedure TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf : string; const Size : Integer)');
17394:  Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Integer');
17395:  Function TCPBufferPeek( var TCPBuf : TTCPBuffer; var Buf: string; const Size:Integer): Integer');
17396:  Function TCPBufferRemove(var TCPBuf : TTCPBuffer;var Buf: string; const Size:Integer): Integer');
17397:  Procedure TCPBufferClear( var TCPBuf : TTCPBuffer)');
17398:  Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer) : Integer');
17399:  Function TCPBufferUsed( const TCPBuf : TTCPBuffer) : Integer');
17400:  Function TCPBufferEmpty( const TCPBuf : TTCPBuffer) : Boolean');
17401:  Function TCPBufferAvailable( const TCPBuf : TTCPBuffer) : Integer');
17402:  Function TCPBufferPtr( const TCPBuf : TTCPBuffer) : Pointer');
17403:  Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer)');
17404: end;
17405:
17406: procedure SIRegister_Glut(CL: TPSPascalCompiler);
17407: begin
17408:   //CL.AddTypeS('PInteger', '^Integer // will not work');
17409:   //CL.AddTypeS('PPChar', '^PChar // will not work');
17410:  CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3);
17411:  CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12);
17412:  CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0);
17413:  CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5);
17414:  CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6);
17415:  Procedure LoadGlut( const dll : String)');
17416:  Procedure FreeGlut');
17417: end;
17418:
17419: procedure SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
17420: begin
17421:   CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17422:  Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean) : THandle');
17423: end;
17424:
17425: procedure SIRegister_SwitchLed(CL: TPSPascalCompiler);
17426: begin
17427:   TLedColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black )');
17428:   TTLedState', '( LedOn, LedOff, LedDisabled )');
17429:   SIRegister_TSwitchLed(CL);
17430:  //CL.AddDelphiFunction('Procedure Register');
17431: end;
17432:
17433: procedure SIRegister_FileClass(CL: TPSPascalCompiler);
17434: begin
17435:   CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17436:    +'xisting, fdOpenAlways, fdTruncateExisting )');
17437:   CL.AddTypeS('TIFileAccess', '( faRead, faWrite, faReadWrite )');
17438:   CL.AddTypeS('TIFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17439:   SIRegister_TCustomFile(CL);
17440:   SIRegister_TIFile(CL);
17441:   SIRegister_TMemoryFile(CL);
17442:   SIRegister_TTextFileReader(CL);
17443:   SIRegister_TTextFileWriter(CL);
17444:   SIRegister_TFileMapping(CL);
```

```
17445:   SIRegister_EFileError(CL);
17446: end;
17447:
17448: procedure SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
17449: begin
17450:   CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17451:    +', ffaDirectory, ffaArchive, ffaAnyFile )');
17452:   CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17453:   SIRegister_TFileSearch(CL);
17454: end;
17455:
17456: procedure SIRegister_uColorFunctions(CL: TPSPascalCompiler);
17457: begin
17458:  TRGBType', 'record RedHex : string; GreenHex : string; BlueHex :'
17459:    + string; Red : integer; Green : integer; Blue : integer; end');
17460:  Function FadeColor( aColor : Longint; aFade : integer) : Tcolor');
17461:  Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17462: end;
17463:
17464: procedure SIRegister_uSettings(CL: TPSPascalCompiler);
17465: begin
17466:  Procedure SaveOscSettings');
17467:  Procedure GetOscSettings');
17468: end;
17469:
17470: procedure SIRegister_cyDebug(CL: TPSPascalCompiler);
17471: begin
17472:   TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer)');
17473:   RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17474:    FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17475:    64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17476:    Cardinal; EnterCount : Integer; ExitCount : Integer; end');
17477:   SIRegister_TcyDebug(CL);
17478: end;
17479:
17480: (*----------------------------------------------------------------------------*)
17481: procedure SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
17482: begin
17483:   TCopyFileResult', '( cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )');
17484:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )');
17485:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )');
17486:   SIRegister_TDestinationOptions(CL);
17487:   TProcCustomCopyFileEvent', 'Procedure ( Sender : TObject; var CopyFileResult : TCopyFileResult)');
17488:   TProcOnCopyFileProgressEvent','Procedure(Sender:TObject;FileBytes,
      TransferedBytes:int64;PercentDone:Int64);
17489:   TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String)');
17490:   SIRegister_TcyCopyFiles(CL);
17491:  Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
      ResetAttr:boolean): TCopyFileResult');
17492:  Function cyCopyFileEx( FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
      TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult');
17493:  Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
       ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;'
17494:                      + FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer');
17495: end;
17496:
17497: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17498: begin
17499:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17500:   SIRegister_TcyFileAttributes(CL);
17501:   SIRegister_TSearchRecInstance(CL);
17502:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17503:   TOptions', 'set of TOption');
17504:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17505:   TProcOnValidateFileEvent','Procedure(Sender:TObject;ValidMaskInclude,ValidMaskExclude,
      ValidAttribs:bool;var Accept:bool;
17506:   TProcOnValidateDirectoryEvent','Procedure(Sender:TObject;Directory:String;var Accept:boolean)');
17507:   SIRegister_TcyCustomSearchFiles(CL);
17508:   SIRegister_TcySearchFiles(CL);
17509:  Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean');
17510:  Function IscyFolder( aSRec : TSearchrec) : Boolean');
17511: end;
17512:
17513:
17514:
17515: {A simple Oscilloscope using  TWaveIn class.
17516:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
17517: uses
17518:   Forms,
17519:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
17520:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
17521:   uColorFunctions in 'uColorFunctions.pas',
17522:   AMixer in 'AMixer.pas',
17523:   uSettings in 'uSettings.pas',
17524:   UWavein4 in 'UWavein4.pas',
17525:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
17526:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
17527:
17528:
```

```
17529: Functions_max hex in the box maXbox
17530: functionslist.txt
17531: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98
17532:
17533: ********************************************************************************
17534: Procedure
17535: PROCEDURE SIZE 8274 8242 7507 7401 6792 6310 5971 4438 3797 3600
17536: Procedure ***************************Now the Procedure list*******************
17537: Procedure ( ACol, ARow : Integer; Items : TStrings)
17538: Procedure ( Agg : TAggregate)
17539: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
17540: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
17541: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
17542: Procedure ( ASender : TObject; const ABytes : Integer)
17543: Procedure ( ASender : TObject; VStream : TStream)
17544: Procedure ( AThread : TIdThread)
17545: Procedure ( AWebModule : TComponent)
17546: Procedure ( Column : TColumn)
17547: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
17548: Procedure ( const iStart : integer; const sText : string)
17549: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
17550: Procedure ( Database : TDatabase; LoginParams : TStrings)
17551: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
17552: Procedure ( DATASET : TDATASET)
17553: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
17554: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
17555: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
17556: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
17557: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
17558: Procedure ( Done : Integer)
17559: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
17560: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
17561: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
17562: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
17563: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
17564: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
17565: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
17566: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
17567: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
17568: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
17569: Procedure ( SENDER : TFIELD; const TEXT : String)
17570: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
17571: Procedure ( Sender : TIdTelnet; const Buffer : String)
17572: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
17573: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
17574: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17575: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
17576: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
17577: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
17578: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
17579: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
17580: Procedure ( Sender : TObject; Button : TMPBtnType)
17581: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
17582: Procedure ( Sender : TObject; Button : TUDBtnType)
17583: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
17584: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
17585: Procedure ( Sender : TObject; Column : TListColumn)
17586: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
17587: Procedure ( Sender : TObject; Connecting : Boolean)
17588: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
17589: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
17590: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
17591: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
17592: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
17593: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
17594: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
17595: Procedure ( Sender : TObject; Index : LongInt)
17596: Procedure ( Sender : TObject; Item : TListItem)
17597: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
17598: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
17599: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
17600: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
17601: Procedure ( Sender : TObject; Item : TListItem; var S : string)
17602: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
17603: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
17604: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
17605: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
17606: Procedure ( Sender : TObject; Node : TTreeNode)
17607: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
17608: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
17609: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
17610: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
17611: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
17612: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
17613: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
17614: Procedure ( Sender : TObject; Rect : TRect)
17615: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
```

```
17616: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
17617: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
17618: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
17619: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
17620: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
17621: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
17622: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
17623: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
17624: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
17625: Procedure ( Sender : TObject; Thread : TServerClientThread)
17626: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
17627: Procedure ( Sender : TObject; Username, Password : string)
17628: Procedure ( Sender : TObject; var AllowChange : Boolean)
17629: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
17630: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
17631: Procedure ( Sender : TObject; var Continue : Boolean)
17632: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
17633: Procedure ( Sender : TObject; var Username : string)
17634: Procedure ( Sender : TObject; Wnd : HWND)
17635: Procedure ( Sender : TToolbar; Button : TToolButton)
17636: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
17637: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
17638: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
17639: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
17640: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
17641: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
17642: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
17643: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
17644: procedure (Sender: TObject)
17645: procedure (Sender: TObject; var Done: Boolean)
17646: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
17647: procedure _T(Name: tbtString; v: Variant);
17648: Procedure AbandonSignalHandler( RtlSigNum : Integer)
17649: Procedure Abort
17650: Procedure About1Click( Sender : TObject)
17651: Procedure Accept( Socket : TSocket)
17652: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
17653: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
17654: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
17655: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
17656: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
17657: Procedure Add( Addend1, Addend2 : TMyBigInt)
17658: Procedure ADD( const AKEY, AVALUE : VARIANT)
17659: Procedure Add( const Key : string; Value : Integer)
17660: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
17661: Procedure ADD( FIELD : TFIELD)
17662: Procedure ADD( ITEM : TMENUITEM)
17663: Procedure ADD( POPUP : TPOPUPMENU)
17664: Procedure AddCharacters( xCharacters : TCharSet)
17665: Procedure AddDriver( const Name : string; List : TStrings)
17666: Procedure AddImages( Value : TCustomImageList)
17667: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
17668: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
17669: Procedure AddLoader( Loader : TBitmapLoader)
17670: Procedure ADDPARAM( VALUE : TPARAM)
17671: Procedure AddPassword( const Password : string)
17672: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
17673: Procedure AddState( oState : TniRegularExpressionState)
17674: Procedure AddStrings( Strings : TStrings);
17675: procedure AddStrings(Strings: TStrings);
17676: Procedure AddStrings1( Strings : TWideStrings);
17677: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
17678: Procedure AddToRecentDocs( const Filename : string)
17679: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
17680: Procedure AllFunctionsList1Click( Sender : TObject)
17681: procedure AllObjectsList1Click(Sender: TObject);
17682: Procedure Allocate( AAllocateBytes : Integer)
17683: procedure AllResourceList1Click(Sender: TObject);
17684: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
17685: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
17686: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
17687: Procedure AnsiFree( var s : AnsiString)
17688: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
17689: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
17690: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
17691: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
17692: Procedure AntiFreeze;
17693: Procedure APPEND
17694: Procedure Append( const S : WideString)
17695: procedure Append(S: string);
17696: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
17697: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
17698: Procedure AppendChunk( Val : OleVariant)
17699: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
17700: Procedure AppendStr( var Dest : string; S : string)
17701: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
17702: Procedure ApplyRange
17703: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17704: Procedure Arrange( Code : TListArrangement)
```

```
17705: procedure Assert(expr : Boolean; const msg: string);
17706: procedure Assert2(expr : Boolean; const msg: string);
17707: Procedure Assign( AList : TCustomBucketList)
17708: Procedure Assign( Other : TObject)
17709: Procedure Assign( Source : TDragObject)
17710: Procedure Assign( Source : TPersistent)
17711: Procedure Assign(Source: TPersistent)
17712: procedure Assign2(mystring, mypath: string);
17713: Procedure AssignCurValues( Source : TDataSet);
17714: Procedure AssignCurValues1( const CurValues : Variant);
17715: Procedure ASSIGNFIELD( FIELD : TFIELD)
17716: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
17717: Procedure AssignFile(var F: Text; FileName: string)
17718: procedure AssignFile(var F: TextFile; FileName: string)
17719: procedure AssignFileRead(var mystring, myfilename: string);
17720: procedure AssignFileWrite(mystring, myfilename: string);
17721: Procedure AssignTo( Other : TObject)
17722: Procedure AssignValues( Value : TParameters)
17723: Procedure ASSIGNVALUES( VALUE : TPARAMS)
17724: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
17725: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
17726: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
17727: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
17728: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17729: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
17730: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
17731: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
17732: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
17733: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
17734: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
17735: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
17736: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
17737: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17738: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
17739: procedure Beep
17740: Procedure BeepOk
17741: Procedure BeepQuestion
17742: Procedure BeepHand
17743: Procedure BeepExclamation
17744: Procedure BeepAsterisk
17745: Procedure BeepInformation
17746: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
17747: Procedure BeginLayout
17748: Procedure BeginTimer( const Delay, Resolution : Cardinal)
17749: Procedure BeginUpdate
17750: procedure BeginUpdate;
17751: procedure BigScreen1Click(Sender: TObject);
17752: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
17753: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
17754: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
17755: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
17756: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
17757: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
17758: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
17759: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
17760: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
17761: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
17762: Procedure BreakPointMenuClick( Sender : TObject)
17763: procedure BRINGTOFRONT
17764: procedure BringToFront;
17765: Procedure btnBackClick( Sender : TObject)
17766: Procedure btnBrowseClick( Sender : TObject)
17767: Procedure BtnClick( Index : TNavigateBtn)
17768: Procedure btnLargeIconsClick( Sender : TObject)
17769: Procedure BuildAndSendRequest( AURI : TIdURI)
17770: Procedure BuildCache
17771: Procedure BurnMemory( var Buff, BuffLen : integer)
17772: Procedure BurnMemoryStream( Destructo : TMemoryStream)
17773: Procedure CalculateFirstSet
17774: Procedure Cancel
17775: procedure CancelDrag;
17776: Procedure CancelEdit
17777: procedure CANCELHINT
17778: Procedure CancelRange
17779: Procedure CancelUpdates
17780: Procedure CancelWriteBuffer
17781: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
17782: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
17783: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
17784: procedure CaptureScreenFormat(vname: string; vextension: string);
17785: procedure CaptureScreenPNG(vname: string);
17786: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
17787: procedure CASCADE
17788: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
17789: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
17790: Procedure cbPathClick( Sender : TObject)
17791: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
17792: Procedure cedebugAfterExecute( Sender : TPSScript)
17793: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
```

```
17794: Procedure cedebugCompile( Sender : TPSScript)
17795: Procedure cedebugExecute( Sender : TPSScript)
17796: Procedure cedebugIdle( Sender : TObject)
17797: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
17798: Procedure CenterHeight( const pc, pcParent : TControl)
17799: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
17800: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
17801: Procedure Change
17802: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
17803: Procedure Changed
17804: Procedure ChangeDir( const ADirName : string)
17805: Procedure ChangeDirUp
17806: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
17807: Procedure ChangeLevelBy( Value : TChangeRange)
17808: Procedure ChDir(const s: string)
17809: Procedure Check(Status: Integer)
17810: Procedure CheckCommonControl( CC : Integer)
17811: Procedure CHECKFIELDNAME( const FIELDNAME : String)
17812: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
17813: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
17814: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
17815: Procedure CheckToken( T : Char)
17816: procedure CheckToken(t:char)
17817: Procedure CheckTokenSymbol( const S : string)
17818: procedure CheckTokenSymbol(s:string)
17819: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
17820: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17821: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
17822: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
17823: procedure CipherFile1Click(Sender: TObject);
17824: Procedure Clear;
17825: Procedure Clear1Click( Sender : TObject)
17826: Procedure ClearColor( Color : TColor)
17827: Procedure CLEARITEM( AITEM : TMENUITEM)
17828: Procedure ClearMapping
17829: Procedure ClearSelection( KeepPrimary : Boolean)
17830: Procedure ClearWriteBuffer
17831: Procedure Click
17832: Procedure Close
17833: Procedure Close1Click( Sender : TObject)
17834: Procedure CloseDatabase( Database : TDatabase)
17835: Procedure CloseDataSets
17836: Procedure CloseDialog
17837: Procedure CloseFile(var F: Text);
17838: Procedure Closure
17839: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17840: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17841: Procedure CodeCompletionList1Click( Sender : TObject)
17842: Procedure ColEnter
17843: Procedure Collapse
17844: Procedure Collapse( Recurse : Boolean)
17845: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
17846: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
17847: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
17848: Procedure Compile1Click( Sender : TObject)
17849: procedure ComponentCount1Click(Sender: TObject);
17850: Procedure Compress(azipfolder, azipfile: string)
17851: Procedure DeCompress(azipfolder, azipfile: string)
17852: Procedure XZip(azipfolder, azipfile: string)
17853: Procedure XUnZip(azipfolder, azipfile: string)
17854: Procedure Connect( const ATimeout : Integer)
17855: Procedure Connect( Socket : TSocket)
17856: procedure Console1Click(Sender: TObject);
17857: Procedure Continue
17858: Procedure ContinueCount( var Counter : TJclCounter)
17859: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
17860: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
17861: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
17862: Procedure ConvertImage(vsource, vdestination: string);
17863: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
17864: Procedure ConvertBitmap(vsource, vdestination: string);
17865: Procedure ConvertToGray(Cnv: TCanvas);
17866: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
17867: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
17868: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
17869: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
17870: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
17871: Procedure CopyFrom( mbCopy : TMyBigInt)
17872: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
17873: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
17874: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
17875: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
17876: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
17877: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
17878: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
17879: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
17880: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
```

```
17881: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17882: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
17883: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17884: Procedure CopyToClipboard
17885: Procedure CountParts
17886: Procedure CreateDataSet
17887: Procedure CreateEmptyFile( const FileName : string)
17888: Procedure CreateFileFromString( const FileName, Data : string)
17889: Procedure CreateFromDelta( Source : TPacketDataSet)
17890: procedure CREATEHANDLE
17891: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
17892: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
17893: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
17894: Procedure CreateTable
17895: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
17896: procedure CSyntax1Click(Sender: TObject);
17897: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
17898: Procedure CURSORPOSCHANGED
17899: procedure CutFirstDirectory(var S: String)
17900: Procedure DataBaseError(const Message: string)
17901: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
17902: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
17903: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
17904: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
17905: Procedure DBIError(errorCode: Integer)
17906: Procedure DebugOutput( const AText : string)
17907: Procedure DebugRun1Click( Sender : TObject)
17908: procedure Dec;
17909: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
17910: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
17911: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
17912: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
17913: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
17914: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
17915: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
17916: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
17917: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
17918: Procedure Decompile1Click( Sender : TObject)
17919: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
17920: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
17921: Procedure DeferLayout
17922: Procedure defFileread
17923: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
17924: Procedure DelayMicroseconds( const MicroSeconds : Integer)
17925: Procedure Delete
17926: Procedure Delete( const AFilename : string)
17927: Procedure Delete( const Index : Integer)
17928: Procedure DELETE( INDEX : INTEGER)
17929: Procedure Delete( Index : LongInt)
17930: Procedure Delete( Node : TTreeNode)
17931: procedure Delete(var s: AnyString; ifrom, icount: Longint);
17932: Procedure DeleteAlias( const Name : string)
17933: Procedure DeleteDriver( const Name : string)
17934: Procedure DeleteIndex( const Name : string)
17935: Procedure DeleteKey( const Section, Ident : String)
17936: Procedure DeleteRecords
17937: Procedure DeleteRecords( AffectRecords : TAffectRecords)
17938: Procedure DeleteString( var pStr : String; const pDelStr : string)
17939: Procedure DeleteTable
17940: procedure DelphiSite1Click(Sender: TObject);
17941: Procedure Deselect
17942: Procedure Deselect( Node : TTreeNode)
17943: procedure DestroyComponents
17944: Procedure DestroyHandle
17945: Procedure Diff( var X : array of Double)
17946: procedure Diff(var X: array of Double);
17947: Procedure DirCreate( const DirectoryName : String)');
17948: procedure DISABLEALIGN
17949: Procedure DisableConstraints
17950: Procedure Disconnect
17951: Procedure Disconnect( Socket : TSocket)
17952: Procedure Dispose
17953: procedure Dispose(P: PChar)
17954: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
17955: Procedure DoKey( Key : TDBCtrlGridKey)
17956: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17957: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17958: Procedure Dormant
17959: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
17960: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
17961: Procedure DoubleToComp( Value : Double; var Result : Comp)
17962: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
17963: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
17964: procedure Draw(X, Y: Integer; Graphic: TGraphic);
17965: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
17966: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17967: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
```

```
17968: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17969: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
17970: procedure DrawFocusRect(const Rect: TRect);
17971: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
17972: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17973: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
17974: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
17975: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
17976: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
17977: Procedure DropConnections
17978: Procedure DropDown
17979: Procedure DumpDescription( oStrings : TStrings)
17980: Procedure DumpStateTable( oStrings : TStrings)
17981: Procedure EDIT
17982: Procedure EditButtonClick
17983: Procedure EditFont1Click( Sender : TObject)
17984: procedure Ellipse(X1, Y1, X2, Y2: Integer);
17985: Procedure Ellipse1( const Rect : TRect);
17986: Procedure EMMS
17987: Procedure Encode( ADest : TStream)
17988: procedure ENDDRAG(DROP:BOOLEAN)
17989: Procedure EndEdit( Cancel : Boolean)
17990: Procedure EndTimer
17991: Procedure EndUpdate
17992: Procedure EraseSection( const Section : string)
17993: Procedure Error( const Ident : string)
17994: procedure Error(Ident:Integer)
17995: Procedure ErrorFmt( const Ident : string; const Args : array of const)
17996: Procedure ErrorStr( const Message : string)
17997: procedure ErrorStr(Message:String)
17998: Procedure Exchange( Index1, Index2 : Integer)
17999: procedure Exchange(Index1, Index2: Integer);
18000: Procedure Exec( FileName, Parameters, Directory : string)
18001: Procedure ExecProc
18002: Procedure ExecSQL( UpdateKind : TUpdateKind)
18003: Procedure Execute
18004: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
18005: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
18006: Procedure ExecuteCommand(executeFile, paramstring: string)
18007: Procedure ExecuteShell(executeFile, paramstring: string)
18008: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18009: Procedure ExitThread(ExitCode: Integer); stdcall;
18010: Procedure ExitProcess(ExitCode: Integer); stdcall;
18011: Procedure Expand( AUserName : String; AResults : TStrings)
18012: Procedure Expand( Recurse : Boolean)
18013: Procedure ExportClipboard1Click( Sender : TObject)
18014: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
18015: Procedure ExtractContentFields( Strings : TStrings)
18016: Procedure ExtractCookieFields( Strings : TStrings)
18017: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
18018: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
18019: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
18020: Procedure ExtractQueryFields( Strings : TStrings)
18021: Procedure FastDegToGrad
18022: Procedure FastDegToRad
18023: Procedure FastGradToDeg
18024: Procedure FastGradToRad
18025: Procedure FastRadToDeg
18026: Procedure FastRadToGrad
18027: Procedure FileClose( Handle : Integer)
18028: Procedure FileClose(handle: integer)
18029: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
18030: Procedure FileStructure( AStructure : TIdFTPDataStructure)
18031: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
18032: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
18033: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
18034: Procedure FillChar2(var X: PChar ; count: integer; value: char)
18035: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
18036: Procedure FillIPList
18037: procedure FillRect(const Rect: TRect);
18038: Procedure FillTStrings( AStrings : TStrings)
18039: Procedure FilterOnBookmarks( Bookmarks : array of const)
18040: procedure FinalizePackage(Module: HMODULE)
18041: procedure FindClose;
18042: procedure FindClose2(var F: TSearchRec)
18043: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
18044: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
18045: Procedure FindNearest( const KeyValues : array of const)
18046: Procedure FinishContext
18047: Procedure FIRST
18048: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
18049: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
18050: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
18051: Procedure FlushSchemaCache( const TableName : string)
18052: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
18053: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
```

```
18054: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
18055: Procedure FormActivate( Sender : TObject)
18056: procedure FormatLn(const format: String; const args: array of const);  //alias
18057: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
18058: Procedure FormCreate( Sender : TObject)
18059: Procedure FormDestroy( Sender : TObject)
18060: Procedure FormKeyPress( Sender : TObject; var Key : Char)
18061: procedure FormOutput1Click(Sender: TObject);
18062: Procedure FormToHtml( Form : TForm; Path : string)
18063: procedure FrameRect(const Rect: TRect);
18064: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
18065: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
18066: Procedure Free( Buffer : TRecordBuffer)
18067: Procedure Free( Buffer : TValueBuffer)
18068: Procedure Free;
18069: Procedure FreeAndNil(var Obj:TObject)
18070: Procedure FreeImage
18071: procedure FreeMem(P: PChar; Size: Integer)
18072: Procedure FreeTreeData( Tree : TUpdateTree)
18073: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
18074: Procedure FullCollapse
18075: Procedure FullExpand
18076: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
18077: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
18078: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
18079: Procedure Get1( AURL : string; const AResponseContent : TStream);
18080: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
18081: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
18082: Procedure GetAliasNames( List : TStrings)
18083: Procedure GetAliasParams( const AliasName : string; List : TStrings)
18084: Procedure GetApplicationsRunning( Strings : TStrings)
18085: Procedure getBox(aURL, extension: string);
18086: Procedure GetCommandTypes( List : TWideStrings)
18087: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
18088: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
18089: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
18090: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
18091: Procedure GetDatabaseNames( List : TStrings)
18092: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
18093: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
18094: Procedure GetDir(d: byte; var s: string)
18095: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
18096: Procedure GetDriverNames( List : TStrings)
18097: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
18098: Procedure GetDriverParams( const DriverName : string; List : TStrings)
18099: Procedure GetEMails1Click( Sender : TObject)
18100: Procedure getEnvironmentInfo;
18101: Function getEnvironmentString: string;
18102: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
18103: Procedure GetFieldNames( const TableName : string; List : TStrings)
18104: Procedure GetFieldNames( const TableName : string; List : TStrings);
18105: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
18106: Procedure GETFIELDNAMES( LIST : TSTRINGS)
18107: Procedure GetFieldNames1( const TableName : string; List : TStrings);
18108: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
18109: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
18110: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
18111: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
18112: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
18113: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
18114: Procedure GetFormatSettings
18115: Procedure GetFromDIB( var DIB : TBitmapInfo)
18116: Procedure GetFromHDIB( HDIB : HBitmap)
18117: Procedure GetIcon( Index : Integer; Image : TIcon);
18118: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
18119: Procedure GetIndexInfo( IndexName : string)
18120: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
18121: Procedure GetIndexNames( List : TStrings)
18122: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
18123: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
18124: Procedure GetIndexNames4( const TableName : string; List : TStrings);
18125: Procedure GetInternalResponse
18126: Procedure GETITEMNAMES( LIST : TSTRINGS)
18127: procedure GetMem(P: PChar; Size: Integer)
18128: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
18129: procedure GetPackageDescription(ModuleName: PChar): string)
18130: Procedure GetPackageNames( List : TStrings);
18131: Procedure GetPackageNames1( List : TWideStrings);
18132: Procedure GetParamList( List : TList; const ParamNames : WideString)
18133: Procedure GetProcedureNames( List : TStrings);
18134: Procedure GetProcedureNames( List : TWideStrings);
18135: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
18136: Procedure GetProcedureNames1( List : TStrings);
18137: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
18138: Procedure GetProcedureNames3( List : TWideStrings);
18139: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
18140: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
18141: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
18142: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
```

```
18143: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
18144: Procedure GetProviderNames( Names : TWideStrings);
18145: Procedure GetProviderNames( Proc : TGetStrProc)
18146: Procedure GetProviderNames1( Names : TStrings);
18147: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
18148: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no open image
18149: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
18150: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
18151: Procedure GetSchemaNames( List : TStrings);
18152: Procedure GetSchemaNames1( List : TWideStrings);
18153: Procedure getScriptandRunAsk;
18154: Procedure getScriptandRun(ascript: string);
18155: Procedure getScript(ascript: string);   //alias
18156: Procedure getWebScript(ascript: string);   //alias
18157: Procedure GetSessionNames( List : TStrings)
18158: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
18159: Procedure GetStrings( List : TStrings)
18160: Procedure GetSystemTime; stdcall;
18161: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
18162: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
18163: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
18164: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
18165: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
18166: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
18167: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
18168: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
18169: Procedure GetVisibleWindows( List : Tstrings)
18170: Procedure GoBegin
18171: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
18172: Procedure GotoCurrent( Table : TTable)
18173: procedure GotoEnd1Click(Sender: TObject);
18174: Procedure GotoNearest
18175: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
18176: Procedure HandleException( E : Exception; var Handled : Boolean)
18177: procedure HANDLEMESSAGE
18178: procedure HandleNeeded;
18179: Procedure Head( AURL : string)
18180: Procedure Help( var AHelpContents : TStringList; ACommand : String)
18181: Procedure HexToBinary( Stream : TStream)
18182: procedure HexToBinary(Stream:TStream)
18183: Procedure HideDragImage
18184: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
18185: Procedure HideTraybar
18186: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
18187: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
18188: Procedure HookOSExceptions
18189: Procedure HookSignal( RtlSigNum : Integer)
18190: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
18191: Procedure HTMLSyntax1Click( Sender : TObject)
18192: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
18193: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
18194: Procedure ImportfromClipboard1Click( Sender : TObject)
18195: Procedure ImportfromClipboard2Click( Sender : TObject)
18196: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
18197: procedure Incb(var x: byte);
18198: Procedure Include1Click( Sender : TObject)
18199: Procedure IncludeOFF;  //preprocessing
18200: Procedure IncludeON;
18201: procedure Info1Click(Sender: TObject);
18202: Procedure InitAltRecBuffers( CheckModified : Boolean)
18203: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
18204: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
18205: Procedure InitData( ASource : TDataSet)
18206: Procedure InitDelta( ADelta : TPacketDataSet);
18207: Procedure InitDelta1( const ADelta : OleVariant);
18208: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
18209: Procedure Initialize
18210: procedure InitializePackage(Module: HMODULE)
18211: Procedure INITIATEACTION
18212: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
18213: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
18214: Procedure InitModule( AModule : TComponent)
18215: Procedure InitStdConvs
18216: Procedure InitTreeData( Tree : TUpdateTree)
18217: Procedure INSERT
18218: Procedure Insert( Index : Integer; AClass : TClass)
18219: Procedure Insert( Index : Integer; AComponent : TComponent)
18220: Procedure Insert( Index : Integer; AObject : TObject)
18221: Procedure Insert( Index : Integer; const S : WideString)
18222: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
18223: Procedure Insert(Index: Integer; const S: string);
18224: procedure Insert(Index: Integer; S: string);
18225: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
18226: procedure InsertComponent(AComponent:TComponent)
18227: procedure InsertControl(AControl: TControl);
18228: Procedure InsertIcon( Index : Integer; Image : TIcon)
18229: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
```

```
18230: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
18231: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
18232: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
18233: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
18234: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
18235: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
18236: Procedure InternalBeforeResolve( Tree : TUpdateTree)
18237: Procedure InvalidateModuleCache
18238: Procedure InvalidateTitles
18239: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
18240: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
18241: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
18242: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
18243: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
18244: procedure JavaSyntax1Click(Sender: TObject);
18245: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
18246: Procedure KillDataChannel
18247: Procedure Largefont1Click( Sender : TObject)
18248: Procedure LAST
18249: Procedure LaunchCpl( FileName : string)
18250: Procedure Launch( const AFile : string)
18251: Procedure LaunchFile( const AFile : string)
18252: Procedure LetFileList(FileList: TStringlist; apath: string);
18253: Procedure lineToNumber( xmemo : String; met : boolean)
18254: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
18255: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
18256: Procedure ListViewData( Sender : TObject; Item : TListItem)
18257: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
18258: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
18259: Procedure ListViewDblClick( Sender : TObject)
18260: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18261: Procedure ListDLLExports(const FileName: string; List: TStrings);
18262: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
18263: procedure LoadBytecode1Click(Sender: TObject);
18264: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
18265: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
18266: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
18267: Procedure LoadFromFile( AFileName : string)
18268: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
18269: Procedure LoadFromFile( const FileName : string)
18270: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
18271: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
18272: Procedure LoadFromFile( const FileName : WideString)
18273: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18274: Procedure LoadFromFile(const AFileName: string)
18275: procedure LoadFromFile(FileName: string);
18276: procedure LoadFromFile(FileName:String)
18277: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
18278: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
18279: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
18280: Procedure LoadFromStream( const Stream : TStream)
18281: Procedure LoadFromStream( S : TStream)
18282: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18283: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18284: Procedure LoadFromStream( Stream : TStream)
18285: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
18286: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
18287: procedure LoadFromStream(Stream: TStream);
18288: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
18289: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
18290: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
18291: Procedure LoadLastFile1Click( Sender : TObject)
18292: { LoadIcoToImage loads two icons from resource named NameRes,
18293:   into two image lists ALarge and ASmall}
18294: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
18295: Procedure LoadMemo
18296: Procedure LoadParamsFromIniFile( FFileName : WideString)
18297: Procedure Lock
18298: Procedure Login
18299: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
18300: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
18301: Procedure MakeCaseInsensitive
18302: Procedure MakeDeterministic( var bChanged : boolean)
18303: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
18304: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
18305: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
18306: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
18307: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
18308: Procedure SetRectComplexFormatStr( const S : string)
18309: Procedure SetPolarComplexFormatStr( const S : string)
18310: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
18311: Procedure MakeVisible
18312: Procedure MakeVisible( PartialOK : Boolean)
18313: Procedure Manual1Click( Sender : TObject)
18314: Procedure MarkReachable
```

```
18315: Procedure maXbox;  //shows the exe version data in a win box
18316: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
18317: Procedure Memo1Change( Sender : TObject)
18318: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
18319: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
18320: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
18321: procedure Memory1Click(Sender: TObject);
18322: Procedure MERGE( MENU : TMAINMENU)
18323: Procedure MergeChangeLog
18324: procedure MINIMIZE
18325: Procedure MinimizeMaxbox;
18326: Procedure MkDir(const s: string)
18327: Procedure MakeDir(const s: string)');
18328: Procedure ChangeDir(const s: string)');
18329: Function makeFile(const FileName: string): integer)');
18330: Procedure mnuPrintFont1Click( Sender : TObject)
18331: procedure ModalStarted
18332: Procedure Modified
18333: Procedure ModifyAlias( Name : string; List : TStrings)
18334: Procedure ModifyDriver( Name : string; List : TStrings)
18335: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
18336: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
18337: Procedure Move( CurIndex, NewIndex : Integer)
18338: procedure Move(CurIndex, NewIndex: Integer);
18339: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
18340: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
18341: Procedure moveCube( o : TMyLabel)
18342: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
18343: procedure MoveTo(X, Y: Integer);
18344: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
18345: Procedure MovePoint(var x,y:Extended; const angle:Extended);
18346: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
18347: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
18348: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
18349: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
18350: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
18351: procedure New(P: PChar)
18352: procedure New1Click(Sender: TObject);
18353: procedure NewInstance1Click(Sender: TObject);
18354: Procedure NEXT
18355: Procedure NextMonth
18356: Procedure Noop
18357: Procedure NormalizePath( var APath : string)
18358: procedure ObjectBinaryToText(Input, Output: TStream)
18359: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18360: procedure ObjectResourceToText(Input, Output: TStream)
18361: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18362: procedure ObjectTextToBinary(Input, Output: TStream)
18363: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18364: procedure ObjectTextToResource(Input, Output: TStream)
18365: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18366: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
18367: Procedure Open( const UserID : WideString; const Password : WideString);
18368: Procedure Open;
18369: Procedure open1Click( Sender : TObject)
18370: Procedure OpenCdDrive
18371: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
18372: Procedure OpenCurrent
18373: Procedure OpenFile(vfilenamepath: string)
18374: Procedure OpenDirectory1Click( Sender : TObject)
18375: Procedure OpenDir(adir: string);
18376: Procedure OpenIndexFile( const IndexName : string)
18377: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
18378: Procedure OpenWriteBuffer( const AThreshhold : Integer)
18379: Procedure OptimizeMem
18380: Procedure Options1( AURL : string);
18381: Procedure OutputDebugString(lpOutputString : PChar)
18382: Procedure PackBuffer
18383: Procedure Paint
18384: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
18385: Procedure PaintToTBitmap( Target : TBitmap)
18386: Procedure PaletteChanged
18387: Procedure ParentBiDiModeChanged
18388: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
18389: Procedure PasteFromClipboard;
18390: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
18391: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
18392: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
18393: Procedure PError( Text : string)
18394: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18395: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
18396: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
18397: procedure playmp3(mpath: string);
18398: Procedure PlayMP31Click( Sender : TObject)
18399: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
18400: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
18401: procedure PolyBezier(const Points: array of TPoint);
```

```
18402: procedure PolyBezierTo(const Points: array of TPoint);
18403: procedure Polygon(const Points: array of TPoint);
18404: procedure Polyline(const Points: array of TPoint);
18405: Procedure Pop
18406: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
18407: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
18408: Procedure POPUP( X, Y : INTEGER)
18409: Procedure PopupURL(URL : WideString);
18410: Procedure POST
18411: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
18412: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
18413: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
18414: Procedure PostUser( const Email, FirstName, LastName : WideString)
18415: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
18416: procedure Pred(X: int64);
18417: Procedure Prepare
18418: Procedure PrepareStatement
18419: Procedure PreProcessXML( AList : TStrings)
18420: Procedure PreventDestruction
18421: Procedure Print( const Caption : string)
18422: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
18423: procedure printf(const format: String; const args: array of const);
18424: Procedure PrintList(Value: TStringList);
18425: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
18426: Procedure Printout1Click( Sender : TObject)
18427: Procedure ProcessHeaders
18428: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
18429: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
18430: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
18431: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
18432: Procedure ProcessMessagesOFF;  //application.processmessages
18433: Procedure ProcessMessagesON;
18434: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
18435: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
18436: Procedure Proclist Size is: 3797 /1415
18437: Procedure procMessClick( Sender : TObject)
18438: Procedure PSScriptCompile( Sender : TPSScript)
18439: Procedure PSScriptExecute( Sender : TPSScript)
18440: Procedure PSScriptLine( Sender : TObject)
18441: Procedure Push( ABoundary : string)
18442: procedure PushItem(AItem: Pointer)
18443: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
18444: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
18445: procedure PutLinuxLines(const Value: string)
18446: Procedure Quit
18447: Procedure RaiseConversionError( const AText : string);
18448: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
18449: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
18450: procedure RaiseException(Ex: TIFException; Param: String);
18451: Procedure RaiseExceptionForLastCmdResult;
18452: procedure RaiseLastException;
18453: procedure RaiseException2;
18454: Procedure RaiseException3(const Msg: string);
18455: Procedure RaiseExcept(const Msg: string);
18456: Procedure RaiseLastOSError
18457: Procedure RaiseLastWin32;
18458: procedure RaiseLastWin32Error)
18459: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
18460: Procedure RandomFillStream( Stream : TMemoryStream)
18461: procedure randomize;
18462: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
18463: Procedure RCS
18464: Procedure Read( Socket : TSocket)
18465: Procedure ReadBlobData
18466: procedure ReadBuffer(Buffer:String;Count:LongInt)
18467: procedure ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
18468: Procedure ReadSection( const Section : string; Strings : TStrings)
18469: Procedure ReadSections( Strings : TStrings)
18470: Procedure ReadSections( Strings : TStrings);
18471: Procedure ReadSections1( const Section : string; Strings : TStrings);
18472: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
18473: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
18474: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
18475: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
18476: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
18477: Procedure Realign;
18478: procedure Rectangle(X1, Y1, X2, Y2: Integer);
18479: Procedure Rectangle1( const Rect : TRect);
18480: Procedure RectCopy( var Dest : TRect; const Source : TRect)
18481: Procedure RectFitToScreen( var R : TRect)
18482: Procedure RectGrow( var R : TRect; const Delta : Integer)
18483: Procedure RectGrowX( var R : TRect; const Delta : Integer)
18484: Procedure RectGrowY( var R : TRect; const Delta : Integer)
18485: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
18486: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
18487: Procedure RectNormalize( var R : TRect)
18488: //  TFileCallbackProcedure = procedure(filename:string);
18489: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
18490: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
```

```
18491: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
18492: Procedure Refresh;
18493: Procedure RefreshData( Options : TFetchOptions)
18494: Procedure REFRESHLOOKUPLIST
18495: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
18496: Procedure RegExPathfinder2(Pathin, fileout, firstp, aregex, ext: string; asort, acopy: boolean);
18497: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
18498: Procedure RegisterChanges( Value : TChangeLink)
18499: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
18500: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
18501: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
18502: Procedure ReInitialize( ADelay : Cardinal)
18503: procedure RELEASE
18504: Procedure Remove( const AByteCount : integer)
18505: Procedure REMOVE( FIELD : TFIELD)
18506: Procedure REMOVE( ITEM : TMENUITEM)
18507: Procedure REMOVE( POPUP : TPOPUPMENU)
18508: Procedure RemoveAllPasswords
18509: procedure RemoveComponent(AComponent:TComponent)
18510: Procedure RemoveDir( const ADirName : string)
18511: Procedure RemoveLambdaTransitions( var bChanged : boolean)
18512: Procedure REMOVEPARAM( VALUE : TPARAM)
18513: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
18514: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
18515: Procedure Rename( const ASourceFile, ADestFile : string)
18516: Procedure Rename( const FileName : string; Reload : Boolean)
18517: Procedure RenameTable( const NewTableName : string)
18518: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
18519: Procedure Replace1Click( Sender : TObject)
18520: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
18521: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
18522: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
18523: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
18524: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
18525: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
18526: Procedure Requery( Options : TExecuteOptions)
18527: Procedure Reset
18528: Procedure Reset1Click( Sender : TObject)
18529: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
18530: procedure ResourceExplore1Click(Sender: TObject);
18531: Procedure RestoreContents
18532: Procedure RestoreDefaults
18533: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
18534: Procedure RetrieveHeaders
18535: Procedure RevertRecord
18536: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
18537: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18538: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18539: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
18540: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
18541: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
18542: Procedure RleCompress2( Stream : TStream)
18543: Procedure RleDecompress2( Stream : TStream)
18544: Procedure RmDir(const S: string)
18545: Procedure Rollback
18546: Procedure Rollback( TransDesc : TTransactionDesc)
18547: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
18548: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
18549: Procedure RollbackTrans
18550: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
18551: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
18552: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
18553: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
18554: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
18555: Procedure S_EBox( const AText : string)
18556: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
18557: Procedure S_IBox( const AText : string)
18558: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
18559: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
18560: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
18561: Procedure SampleVarianceAndMean
18562: ( const X : TDynFloatArray; var Variance, Mean : Float)
18563: Procedure Save2Click( Sender : TObject)
18564: Procedure Saveas3Click( Sender : TObject)
18565: Procedure Savebefore1Click( Sender : TObject)
18566: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
18567: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
18568: Procedure SaveConfigFile
18569: Procedure SaveOutput1Click( Sender : TObject)
18570: procedure SaveScreenshotClick(Sender: TObject);
18571: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
18572: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
18573: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
18574: Procedure SaveToFile( AFileName : string)
18575: Procedure SAVETOFILE( const FILENAME : String)
18576: Procedure SaveToFile( const FileName : WideString)
18577: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
18578: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
```

```
18579: procedure SaveToFile(FileName: string);
18580: procedure SaveToFile(FileName:String)
18581: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
18582: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18583: Procedure SaveToStream( S : TStream)
18584: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18585: Procedure SaveToStream( Stream : TStream)
18586: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
18587: procedure SaveToStream(Stream: TStream);
18588: procedure SaveToStream(Stream:TStream)
18589: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
18590: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
18591: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
18592: procedure Say(const sText: string)
18593: Procedure SBytecode1Click( Sender : TObject)
18594: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
18595: procedure ScriptExplorer1Click(Sender: TObject);
18596: Procedure Scroll( Distance : Integer)
18597: Procedure Scroll( DX, DY : Integer)
18598: procedure ScrollBy(DeltaX, DeltaY: Integer);
18599: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
18600: Procedure ScrollTabs( Delta : Integer)
18601: Procedure Search1Click( Sender : TObject)
18602: procedure SearchAndOpenDoc(vfilenamepath: string)
18603: procedure SearchAndOpenFile(vfilenamepath: string)
18604: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
18605: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
18606: Procedure SearchNext1Click( Sender : TObject)
18607: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
18608: Procedure Select1( const Nodes : array of TTreeNode);
18609: Procedure Select2( Nodes : TList);
18610: Procedure SelectNext( Direction : Boolean)
18611: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
18612: Procedure SelfTestPEM  //unit uPSI_cPEM
18613: Procedure Send( AMsg : TIdMessage)
18614: //config forst in const MAILINIFILE = 'maildef.ini';
18615: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
18616: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18617: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18618: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
18619: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
18620: Procedure SendResponse
18621: Procedure SendStream( AStream : TStream)
18622: Procedure Set8087CW( NewCW : Word)
18623: Procedure SetAll( One, Two, Three, Four : Byte)
18624: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
18625: Procedure SetAppDispatcher( const ADispatcher : TComponent)
18626: procedure SetArrayLength;
18627: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
18628: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
18629: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer);  //all init
18630: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
18631: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);');
18632: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);');
18633: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
18634: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
18635: Procedure SetAsHandle( Format : Word; Value : THandle)
18636: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
18637: procedure SetCaptureControl(Control: TControl);
18638: Procedure SetColumnAttributes
18639: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
18640: Procedure SetCustomHeader( const Name, Value : string)
18641: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
18642: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
18643: Procedure SetFocus
18644: procedure SetFocus; virtual;
18645: Procedure SetInitialState
18646: Procedure SetKey
18647: procedure SetLastError(ErrorCode: Integer)
18648: procedure SetLength;
18649: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
18650: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
18651: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
18652: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
18653: Procedure SetParams1( UpdateKind : TUpdateKind);
18654: Procedure SetPassword( const Password : string)
18655: Procedure SetPointer( Ptr : Pointer; Size : Longint)
18656: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
18657: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
18658: Procedure SetProvider( Provider : TComponent)
18659: Procedure SetProxy( const Proxy : string)
18660: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
18661: Procedure SetRange( const StartValues, EndValues : array of const)
18662: Procedure SetRangeEnd
18663: Procedure SetRate( const aPercent, aYear : integer)
18664: procedure SetRate(const aPercent, aYear: integer)
18665: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
18666: Procedure SetSafeCallExceptionMsg( Msg : String)
```

```
18667: procedure SETSELTEXTBUF(BUFFER:PCHAR)
18668: Procedure SetSize( AWidth, AHeight : Integer)
18669: procedure SetSize(NewSize:LongInt)
18670: procedure SetString(var s: string; buffer: PChar; len: Integer)
18671: Procedure SetStrings( List : TStrings)
18672: Procedure SetText( Text : PwideChar)
18673: procedure SetText(Text: PChar);
18674: Procedure SetTextBuf( Buffer : PChar)
18675: procedure SETTEXTBUF(BUFFER:PCHAR)
18676: Procedure SetTick( Value : Integer)
18677: Procedure SetTimeout( ATimeOut : Integer)
18678: Procedure SetTraceEvent( Event : TDBXTraceEvent)
18679: Procedure SetUserName( const UserName : string)
18680: Procedure SetWallpaper( Path : string);
18681: procedure ShellStyle1Click(Sender: TObject);
18682: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
18683: Procedure ShowFileProperties( const FileName : string)
18684: Procedure ShowInclude1Click( Sender : TObject)
18685: Procedure ShowInterfaces1Click( Sender : TObject)
18686: Procedure ShowLastException1Click( Sender : TObject)
18687: Procedure ShowMessage( const Msg : string)
18688: Procedure ShowMessageBig(const aText : string);
18689: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
18690: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
18691: Procedure MsgBig(const aText : string);        //alias
18692: procedure showmessage(mytext: string);
18693: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
18694: procedure ShowMessageFmt(const Msg: string; Params: array of const))
18695: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
18696: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
18697: Procedure ShowSearchDialog( const Directory : string)
18698: Procedure ShowSpecChars1Click( Sender : TObject)
18699: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
18700: Procedure ShredFile( const FileName : string; Times : Integer)
18701: procedure Shuffle(vQ: TStringList);
18702: Procedure ShuffleList( var List : array of Integer; Count : Integer)
18703: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
18704: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
18705: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
18706: Procedure Site( const ACommand : string)
18707: Procedure SkipEOL
18708: Procedure Sleep( ATime : cardinal)
18709: Procedure Sleep( milliseconds : Cardinal)
18710: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
18711: Procedure Slinenumbers1Click( Sender : TObject)
18712: Procedure Sort
18713: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
18714: procedure Speak(const sText: string)   //async like voice
18715: procedure Speak2(const sText: string)  //sync
18716: procedure Split(Str: string;  SubStr: string; List: TStrings);
18717: Procedure SplitNameValue( const Line : string; var Name, Value : string)
18718: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
18719: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
18720: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
18721: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
18722: procedure SQLSyntax1Click(Sender: TObject);
18723: Procedure SRand( Seed : RNG_IntType)
18724: Procedure Start
18725: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
18726: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
18727: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
18728: Procedure StartTransaction( TransDesc : TTransactionDesc)
18729: Procedure Status( var AStatusList : TStringList)
18730: Procedure StatusBar1DblClick( Sender : TObject)
18731: Procedure StepInto1Click( Sender : TObject)
18732: Procedure StepIt
18733: Procedure StepOut1Click( Sender : TObject)
18734: Procedure Stop
18735: procedure stopmp3;
18736: procedure StartWeb(aurl: string);
18737: Procedure Str(aint: integer; astr: string); //of system
18738: Procedure StrDispose( Str : PChar)
18739: procedure StrDispose(Str: PChar)
18740: Procedure StrReplace(var Str: String; Old, New: String);
18741: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
18742: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
18743: Procedure StringToBytes( Value : String; Bytes : array of byte)
18744: procedure StrSet(c : Char; I : Integer; var s : String);
18745: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
18746: Procedure StructureMount( APath : String)
18747: procedure STYLECHANGED(SENDER:TOBJECT)
18748: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
18749: procedure Succ(X: int64);
18750: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
18751: procedure SwapChar(var X,Y: char);  //swapX follows
18752: Procedure SwapFloats( var X, Y : Float)
18753: procedure SwapGrid(grd: TStringGrid);
18754: Procedure SwapOrd( var I, J : Byte);
18755: Procedure SwapOrd( var X, Y : Integer)
```

```
18756: Procedure SwapOrd1( var I, J : Shortint);
18757: Procedure SwapOrd2( var I, J : Smallint);
18758: Procedure SwapOrd3( var I, J : Word);
18759: Procedure SwapOrd4( var I, J : Integer);
18760: Procedure SwapOrd5( var I, J : Cardinal);
18761: Procedure SwapOrd6( var I, J : Int64);
18762: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
18763: Procedure Synchronize1( Method : TMethod);
18764: procedure SyntaxCheck1Click(Sender: TObject);
18765: Procedure SysFreeString(const S: WideString); stdcall;
18766: Procedure TakeOver( Other : TLinearBitmap)
18767: Procedure Talkln(const sText: string)  //async voice
18768: procedure tbtn6resClick(Sender: TObject);
18769: Procedure tbtnUseCaseClick( Sender : TObject)
18770: procedure TerminalStyle1Click(Sender: TObject);
18771: Procedure Terminate
18772: Procedure texSyntax1Click( Sender : TObject)
18773: procedure TextOut(X, Y: Integer; Text: string);
18774: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
18775: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
18776: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
18777: Procedure TextStart
18778: procedure TILE
18779: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
18780: Procedure TitleClick( Column : TColumn)
18781: Procedure ToDo
18782: procedure toolbtnTutorialClick(Sender: TObject);
18783: Procedure Trace1( AURL : string; const AResponseContent : TStream);
18784: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
18785: Procedure Truncate
18786: procedure Tutorial101Click(Sender: TObject);
18787: procedure Tutorial10Statistics1Click(Sender: TObject);
18788: procedure Tutorial11Forms1Click(Sender: TObject);
18789: procedure Tutorial12SQL1Click(Sender: TObject);
18790: Procedure tutorial1Click( Sender : TObject)
18791: Procedure tutorial21Click( Sender : TObject)
18792: Procedure tutorial31Click( Sender : TObject)
18793: Procedure tutorial4Click( Sender : TObject)
18794: Procedure Tutorial5Click( Sender : TObject)
18795: procedure Tutorial6Click(Sender: TObject);
18796: procedure Tutorial91Click(Sender: TObject);
18797: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
18798: procedure UniqueString(var str: AnsiString)
18799: procedure UnloadLoadPackage(Module: HMODULE)
18800: Procedure Unlock
18801: Procedure UNMERGE( MENU : TMAINMENU)
18802: Procedure UnRegisterChanges( Value : TChangeLink)
18803: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
18804: Procedure UnregisterConversionType( const AType : TConvType)
18805: Procedure UnRegisterProvider( Prov : TCustomProvider)
18806: Procedure UPDATE
18807: Procedure UpdateBatch( AffectRecords : TAffectRecords)
18808: Procedure UPDATECURSORPOS
18809: Procedure UpdateFile
18810: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
18811: Procedure UpdateResponse( AResponse : TWebResponse)
18812: Procedure UpdateScrollBar
18813: Procedure UpdateView1Click( Sender : TObject)
18814: procedure Val(const s: string; var n, z: Integer)
18815: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
18816: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
18817: Procedure VariantAdd( const src : Variant; var dst : Variant)
18818: Procedure VariantAnd( const src : Variant; var dst : Variant)
18819: Procedure VariantArrayRedim( var V : Variant; High : Integer)
18820: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
18821: Procedure VariantClear( var V : Variant)
18822: Procedure VariantCpy( const src : Variant; var dst : Variant)
18823: Procedure VariantDiv( const src : Variant; var dst : Variant)
18824: Procedure VariantMod( const src : Variant; var dst : Variant)
18825: Procedure VariantMul( const src : Variant; var dst : Variant)
18826: Procedure VariantOr( const src : Variant; var dst : Variant)
18827: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
18828: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
18829: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
18830: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
18831: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
18832: Procedure VariantShl( const src : Variant; var dst : Variant)
18833: Procedure VariantShr( const src : Variant; var dst : Variant)
18834: Procedure VariantSub( const src : Variant; var dst : Variant)
18835: Procedure VariantXor( const src : Variant; var dst : Variant)
18836: Procedure VarCastError;
18837: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
18838: Procedure VarInvalidOp
18839: Procedure VarInvalidNullOp
18840: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
18841: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
18842: Procedure VarArrayCreateError
18843: Procedure VarResultCheck( AResult : HRESULT);
18844: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
```

```
18845: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
18846: Function VarTypeAsText( const AType : TVarType) : string
18847: procedure Voice(const sText: string) //async
18848: procedure Voice2(const sText: string) //sync
18849: Procedure WaitMiliSeconds( AMSec : word)
18850: Procedure WaitMS( AMSec : word)');
18851: procedure WebCamPic(picname: string);  //eg: c:\mypic.png
18852: Procedure WideAppend( var dst : WideString; const src : WideString)
18853: Procedure WideAssign( var dst : WideString; var src : WideString)
18854: Procedure WideDelete( var dst : WideString; index, count : Integer)
18855: Procedure WideFree( var s : WideString)
18856: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
18857: Procedure WideFromPChar( var dst : WideString; src : PChar)
18858: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
18859: Procedure WideSetLength( var dst : WideString; len : Integer)
18860: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
18861: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
18862: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
18863: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
18864: Procedure HttpGet(const Url: string; Stream:TStream);
18865: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
18866: Procedure WordWrap1Click( Sender : TObject)
18867: Procedure Write( const AOut : string)
18868: Procedure Write( Socket : TSocket)
18869: procedure Write(S: string);
18870: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
18871: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
18872: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
18873: procedure WriteBuffer(Buffer:String;Count:LongInt)
18874: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
18875: Procedure WriteChar( AValue : Char)
18876: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
18877: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
18878: Procedure WriteFloat( const Section, Name : string; Value : Double)
18879: Procedure WriteHeader( AHeader : TStrings)
18880: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
18881: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
18882: Procedure WriteLn( const AOut : string)
18883: procedure Writeln(s: string);
18884: Procedure WriteLog( const FileName, LogLine : string)
18885: Procedure WriteRFCReply( AReply : TIdRFCReply)
18886: Procedure WriteRFCStrings( AStrings : TStrings)
18887: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
18888: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
18889: Procedure WriteString( const Section, Ident, Value : String)
18890: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
18891: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
18892: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
18893: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18894: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18895: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
18896: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
18897: procedure XMLSyntax1Click(Sender: TObject);
18898: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
18899: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
18900: Procedure ZeroFillStream( Stream : TMemoryStream)
18901: procedure XMLSyntax1Click(Sender: TObject);
18902: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
18903: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
18904: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
18905: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
18906: procedure(Sender, Source: TObject;X, Y: Integer)
18907: procedure(Sender, Target: TObject; X, Y: Integer)
18908: procedure(Sender: TObject; ASection, AWidth: Integer)
18909: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
18910: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
18911: procedure(Sender: TObject; var Action: TCloseAction)
18912: procedure(Sender: TObject; var CanClose: Boolean)
18913: procedure(Sender: TObject; var Key: Char);
18914: ProcedureName ProcedureNames ProcedureParametersCursor @
18915:
18916: *************Now Constructors constructor *************
18917:  Size is: 1248 1115 996 628 550 544 501 459 (381)
18918:  Attach( VersionInfoData : Pointer; Size : Integer)
18919:  constructor Create( ABuckets : TBucketListSizes)
18920:  Create( ACallBackWnd : HWND)
18921:  Create( AClient : TCustomTaskDialog)
18922:  Create( AClient : TIdTelnet)
18923:  Create( ACollection : TCollection)
18924:  Create( ACollection : TFavoriteLinkItems)
18925:  Create( ACollection : TTaskDialogButtons)
18926:  Create( AConnection : TIdCustomHTTP)
18927:  Create( ACreateSuspended : Boolean)
18928:  Create( ADataSet : TCustomSQLDataSet)
18929:  CREATE( ADATASET : TDATASET)
18930:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
18931:  Create( AGrid : TCustomDBGrid)
18932:  Create( AGrid : TStringGrid; AIndex : Longint)
18933:  Create( AHTTP : TIdCustomHTTP)
```

```
18934:  Create( AListItems : TListItems)
18935:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
18936:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
18937:  Create( AOwner : TCommonCalendar)
18938:  Create( AOwner : TComponent)
18939:  CREATE( AOWNER : TCOMPONENT)
18940:  Create( AOwner : TCustomListView)
18941:  Create( AOwner : TCustomOutline)
18942:  Create( AOwner : TCustomRichEdit)
18943:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
18944:  Create( AOwner : TCustomTreeView)
18945:  Create( AOwner : TIdUserManager)
18946:  Create( AOwner : TListItems)
18947:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
18948:  CREATE( AOWNER : TPERSISTENT)
18949:  Create( AOwner : TPersistent)
18950:  Create( AOwner : TTable)
18951:  Create( AOwner : TTreeNodes)
18952:  Create( AOwner : TWinControl; const ClassName : string)
18953:  Create( AParent : TIdCustomHTTP)
18954:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
18955:  Create( AProvider : TBaseProvider)
18956:  Create( AProvider : TCustomProvider);
18957:  Create( AProvider : TDataSetProvider)
18958:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
18959:  Create( ASocket : TSocket)
18960:  Create( AStrings : TWideStrings)
18961:  Create( AToolBar : TToolBar)
18962:  Create( ATreeNodes : TTreeNodes)
18963:  Create( Autofill : boolean)
18964:  Create( AWebPageInfo : TAbstractWebPageInfo)
18965:  Create( AWebRequest : TWebRequest)
18966:  Create( Collection : TCollection)
18967:  Create( Collection : TIdMessageParts; ABody : TStrings)
18968:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
18969:  Create( Column : TColumn)
18970:  Create( const AConvFamily : TConvFamily; const ADescription : string)
18971:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
18972:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
18973:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
18974:  Create( const ATabSet : TTabSet)
18975:  Create( const Compensate : Boolean)
18976:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
18977:  Create( const FileName : string)
18978:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
18979:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
18980:  Create( const MaskValue : string)
18981:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
18982:  Create( const Prefix : string)
18983:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
18984:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18985:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18986:  Create( CoolBar : TCoolBar)
18987:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
18988:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
18989:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
18990:  Create( DBCtrlGrid : TDBCtrlGrid)
18991:  Create( DSTableProducer : TDSTableProducer)
18992:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
18993:  Create( ErrorCode : DBIResult)
18994:  Create( Field : TBlobField; Mode : TBlobStreamMode)
18995:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
18996:  Create( HeaderControl : TCustomHeaderControl)
18997:  Create( HTTPRequest : TWebRequest)
18998:  Create( iStart : integer; sText : string)
18999:  Create( iValue : Integer)
19000:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
19001:  Create( MciErrNo : MCIERROR; const Msg : string)
19002:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
19003:  Create( Message : string; ErrorCode : DBResult)
19004:  Create( Msg : string)
19005:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
19006:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
19007:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
19008:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
19009:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
19010:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
19011:  Create( Owner : TCustomComboBoxEx)
19012:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
19013:  Create( Owner : TPersistent)
19014:  Create( Params : TStrings)
19015:  Create( Size : Cardinal)
19016:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
19017:  Create( StatusBar : TCustomStatusBar)
19018:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
```

```
19019:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
19020:  Create(AHandle:Integer)
19021:  Create(AOwner: TComponent); virtual;
19022:  Create(const AURI : string)
19023:  Create(FileName:String;Mode:Word)
19024:  Create(Instance:THandle;ResName:String;ResType:PChar)
19025:  Create(Stream : TStream)
19026:  Create1( ADataset : TDataset);
19027:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
19028:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
19029:  Create2( Other : TObject);
19030:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
19031:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
19032:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
19033:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
19034:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
19035:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
19036:  CreateRes( Ident : Integer);
19037:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
19038:  CreateRes( ResStringRec : PResStringRec);
19039:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
19040:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
19041:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
19042:  CreateSize( AWidth, AHeight : Integer)
19043:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
19044:
19045: ------------------------------------------------------------------------------
19046: unit uPSI_MathMax;
19047: ------------------------------------------------------------------------------
19048:  CONSTS
19049:   Bernstein: Float = 0.28016949902386913303643649123 07;  // Bernstein constant
19050:   Cbrt2: Float     = 1.25992104989487316476721060727 82;  // CubeRoot(2)
19051:   Cbrt3: Float     = 1.44224957030740838232163831078 01;  // CubeRoot(3)
19052:   Cbrt10: Float    = 2.15443469003188372175929356651 94;  // CubeRoot(10)
19053:   Cbrt100: Float   = 4.64158883361277889241007635091 94;  // CubeRoot(100)
19054:   CbrtPi: Float    = 1.46459188756152326302014252726 38;  // CubeRoot(PI)
19055:   Catalan: Float   = 0.91596559417721901505460351493 24;  // Catalan constant
19056:   PiJ:    Float      = 3.14159265358979323846264338327 95;  // PI
19057:   PI:  Extended =  3.14159265358979323846264338327 95);
19058:   PiOn2: Float     = 1.57079632679489661923132169163 98;  // PI / 2
19059:   PiOn3: Float     = 1.04719755119659774615421446109 32;  // PI / 3
19060:   PiOn4: Float     = 0.78539816339744830961566084581 988; // PI / 4
19061:   Sqrt2: Float     = 1.41421356237309504880168872420 97;  // Sqrt(2)
19062:   Sqrt3: Float     = 1.73205080756887729352744634150 59;  // Sqrt(3)
19063:   Sqrt5: Float     = 2.23606797749978969640917366873 13;  // Sqrt(5)
19064:   Sqrt10: Float    = 3.16227766016837933199889354443 27;  // Sqrt(10)
19065:   SqrtPi: Float    = 1.77245385090551602729816748334 11;  // Sqrt(PI)
19066:   Sqrt2Pi: Float   = 2.50662827463100050241576528481 1;   // Sqrt(2 * PI)
19067:   TwoPi: Float     = 6.28318530717958647692528676655 9;   // 2 * PI
19068:   ThreePi: Float   = 9.42477796076937971538793014983 85;  // 3 * PI
19069:   Ln2: Float       = 0.69314718055994530941723212145818; // Ln(2)
19070:   Ln10: Float      = 2.30258509299404568401799145468 44;  // Ln(10)
19071:   LnPi: Float      = 1.14472988584940017414342735135 31;  // Ln(PI)
19072:   Log2J: Float     = 0.30102999566398119521373889472449; // Log10(2)
19073:   Log3: Float      = 0.47712125471966243729502790325512; // Log10(3)
19074:   LogPi: Float     = 0.49714987269413385435126828829 09;  // Log10(PI)
19075:   LogE: Float      = 0.43429448190325182765112891891661; // Log10(E)
19076:   E: Float         = 2.71828182845904523536028747135 27;  // Natural constant
19077:   hLn2Pi: Float    = 0.91893853320467274178032973640562; // Ln(2*PI)/2
19078:   inv2Pi: Float    = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
19079:   TwoToPower63: Float = 9223372036854775808.0;          // 2^63
19080:   GoldenMean: Float   = 1.61803398874989484820458683436563 8;  // GoldenMean
19081:   EulerMascheroni: Float = 0.57721566490153286060651209008 24;  // Euler GAMMA
19082:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
19083:   StDelta         : Extended = 0.00001;  {delta for difference equations}
19084:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
19085:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
19086:
19087: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
19088: begin
19089:   MetersPerInch = 0.0254; // [1]
19090:   MetersPerFoot = MetersPerInch * 12;
19091:   MetersPerYard = MetersPerFoot * 3;
19092:   MetersPerMile = MetersPerFoot * 5280;
19093:   MetersPerNauticalMiles = 1852;
19094:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
19095:   MetersPerLightSecond = 2.99792458E8; // [5]
19096:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
19097:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
19098:   MetersPerCubit = 0.4572; // [6][7]
19099:   MetersPerFathom = MetersPerFoot * 6;
19100:   MetersPerFurlong = MetersPerYard * 220;
19101:   MetersPerHand = MetersPerInch * 4;
19102:   MetersPerPace = MetersPerInch * 30;
19103:   MetersPerRod = MetersPerFoot * 16.5;
19104:   MetersPerChain = MetersPerRod * 4;
19105:   MetersPerLink = MetersPerChain / 100;
19106:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
```

```
19107:   MetersPerPica = MetersPerPoint * 12;
19108:
19109:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
19110:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
19111:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
19112:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
19113:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
19114:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
19115:
19116:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
19117:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
19118:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
19119:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
19120:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
19121:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
19122:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
19123:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
19124:
19125:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
19126:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
19127:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
19128:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
19129:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
19130:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
19131:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
19132:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
19133:
19134:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
19135:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
19136:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
19137:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
19138:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
19139:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
19140:
19141:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
19142:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
19143:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
19144:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
19145:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
19146:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
19147:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
19148:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
19149:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
19150:
19151:   GramsPerPound = 453.59237; // [1][7]
19152:   GramsPerDrams = GramsPerPound / 256;
19153:   GramsPerGrains = GramsPerPound / 7000;
19154:   GramsPerTons = GramsPerPound * 2000;
19155:   GramsPerLongTons = GramsPerPound * 2240;
19156:   GramsPerOunces = GramsPerPound / 16;
19157:   GramsPerStones = GramsPerPound * 14;
19158:
19159:   MaxAngle 9223372036854775808.0;
19160:   MaxTanH 5678.26170314707197474596553898 54);
19161:   MaxFactorial( 1754);
19162:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
19163:   MinFloatingPoint',(3.362103143112093506262677817 3218E-4932);
19164:   MaxTanH( 354.89135644669199842162284618659);
19165:   MaxFactorial'LongInt'( 170);
19166:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
19167:   MinFloatingPointD(2.225073858507201383090232717 3324E-308);
19168:   MaxTanH( 44.361419555836499802702855773323);
19169:   MaxFactorial'LongInt'( 33);
19170:   MaxFloatingPointS( 3.402823669209384634633746074 3177E+38);
19171:   MinFloatingPointS( 1.175494350822287507968736537 2222E-38);
19172:   PiExt( 3.14159265358979323846264338327 95);
19173:   RatioDegToRad( PiExt / 180.0);
19174:   RatioGradToRad( PiExt / 200.0);
19175:   RatioDegToGrad( 200.0 / 180.0);
19176:   RatioGradToDeg( 180.0 / 200.0);
19177:   Crc16PolynomCCITT'LongWord $1021);
19178:  Crc16PolynomIBM'LongWord $8005);
19179:  Crc16Bits'LongInt'( 16);
19180:  Crc16Bytes'LongInt'( 2);
19181:  Crc16HighBit'LongWord $8000);
19182:  NotCrc16HighBit','LongWord $7FFF);
19183:   Crc32PolynomIEEE','LongWord $04C11DB7);
19184:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
19185:  Crc32Koopman','LongWord $741B8CD7);
19186:  Crc32Bits','LongInt'( 32);
19187:  Crc32Bytes','LongInt'( 4);
19188:  Crc32HighBit','LongWord $80000000);
19189:  NotCrc32HighBit','LongWord $7FFFFFFF);
19190:
19191:   MinByte       = Low(Byte);
19192:   MaxByte       = High(Byte);
19193:   MinWord       = Low(Word);
19194:   MaxWord       = High(Word);
19195:   MinShortInt   = Low(ShortInt);
```

```
19196:   MaxShortInt   = High(ShortInt);
19197:   MinSmallInt   = Low(SmallInt);
19198:   MaxSmallInt   = High(SmallInt);
19199:   MinLongWord   = LongWord(Low(LongWord));
19200:   MaxLongWord   = LongWord(High(LongWord));
19201:   MinLongInt    = LongInt(Low(LongInt));
19202:   MaxLongInt    = LongInt(High(LongInt));
19203:   MinInt64      = Int64(Low(Int64));
19204:   MaxInt64      = Int64(High(Int64));
19205:   MinInteger    = Integer(Low(Integer));
19206:   MaxInteger    = Integer(High(Integer));
19207:   MinCardinal   = Cardinal(Low(Cardinal));
19208:   MaxCardinal   = Cardinal(High(Cardinal));
19209:   MinNativeUInt = NativeUInt(Low(NativeUInt));
19210:   MaxNativeUInt = NativeUInt(High(NativeUInt));
19211:   MinNativeInt  = NativeInt(Low(NativeInt));
19212:   MaxNativeInt  = NativeInt(High(NativeInt));
19213:  Function CosH( const Z : Float) : Float;
19214:  Function SinH( const Z : Float) : Float;
19215:  Function TanH( const Z : Float) : Float;
19216:
19217:
19218:   //**********from DMath.Dll Lib of types.inc in source\dmath_dll
19219:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
19220:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
19221:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
19222:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
19223:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
19224:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
19225:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
19226:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
19227:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
19228:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
19229:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
19230:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
19231:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
19232:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
19233:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
19234:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
19235:   MaxLog = 709.7827128933840;
19236:   MinLog = -708.3964185322641;
19237:   MaxFac = 170;
19238:   MaxGam = 171.624376956302;
19239:   MaxLgm = 2.556348E+305;
19240:   SingleCompareDelta   = 1.0E-34;
19241:   DoubleCompareDelta   = 1.0E-280;
19242:   {$IFDEF CLR}
19243:   ExtendedCompareDelta = DoubleCompareDelta;
19244:   {$ELSE}
19245:   ExtendedCompareDelta = 1.0E-4400;
19246:   {$ENDIF}
19247:   Bytes1KB  = 1024;
19248:   Bytes1MB  = 1024 * Bytes1KB;
19249:   Bytes1GB  = 1024 * Bytes1MB;
19250:   Bytes64KB = 64 * Bytes1KB;
19251:   Bytes64MB = 64 * Bytes1MB;
19252:   Bytes2GB  = 2 * LongWord(Bytes1GB);
19253:     clBlack32', $FF000000 ));
19254:     clDimGray32', $FF3F3F3F ));
19255:     clGray32', $FF7F7F7F ));
19256:     clLightGray32', $FFBFBFBF ));
19257:     clWhite32', $FFFFFFFF ));
19258:     clMaroon32', $FF7F0000 ));
19259:     clGreen32', $FF007F00 ));
19260:     clOlive32', $FF7F7F00 ));
19261:     clNavy32', $FF00007F ));
19262:     clPurple32', $FF7F007F ));
19263:     clTeal32', $FF007F7F ));
19264:     clRed32', $FFFF0000 ));
19265:     clLime32', $FF00FF00 ));
19266:     clYellow32', $FFFFFF00 ));
19267:     clBlue32', $FF0000FF ));
19268:     clFuchsia32', $FFFF00FF ));
19269:     clAqua32', $FF00FFFF ));
19270:     clAliceBlue32', $FFF0F8FF ));
19271:     clAntiqueWhite32', $FFFAEBD7 ));
19272:     clAquamarine32', $FF7FFFD4 ));
19273:     clAzure32', $FFF0FFFF ));
19274:     clBeige32', $FFF5F5DC ));
19275:     clBisque32', $FFFFE4C4 ));
19276:     clBlancheDalmond32', $FFFFEBCD ));
19277:     clBlueViolet32', $FF8A2BE2 ));
19278:     clBrown32', $FFA52A2A ));
19279:     clBurlyWood32', $FFDEB887 ));
19280:     clCadetblue32', $FF5F9EA0 ));
19281:     clChartReuse32', $FF7FFF00 ));
19282:     clChocolate32', $FFD2691E ));
19283:     clCoral32', $FFFF7F50 ));
19284:     clCornFlowerBlue32', $FF6495ED ));
```

```
19285:     clCornSilk32', $FFFFF8DC ));
19286:     clCrimson32', $FFDC143C ));
19287:     clDarkBlue32', $FF00008B ));
19288:     clDarkCyan32', $FF008B8B ));
19289:     clDarkGoldenRod32', $FFB8860B ));
19290:     clDarkGray32', $FFA9A9A9 ));
19291:     clDarkGreen32', $FF006400 ));
19292:     clDarkGrey32', $FFA9A9A9 ));
19293:     clDarkKhaki32', $FFBDB76B ));
19294:     clDarkMagenta32', $FF8B008B ));
19295:     clDarkOliveGreen32', $FF556B2F ));
19296:     clDarkOrange32', $FFFF8C00 ));
19297:     clDarkOrchid32', $FF9932CC ));
19298:     clDarkRed32', $FF8B0000 ));
19299:     clDarkSalmon32', $FFE9967A ));
19300:     clDarkSeaGreen32', $FF8FBC8F ));
19301:     clDarkSlateBlue32', $FF483D8B ));
19302:     clDarkSlateGray32', $FF2F4F4F ));
19303:     clDarkSlateGrey32', $FF2F4F4F ));
19304:     clDarkTurquoise32', $FF00CED1 ));
19305:     clDarkViolet32', $FF9400D3 ));
19306:     clDeepPink32', $FFFF1493 ));
19307:     clDeepSkyBlue32', $FF00BFFF ));
19308:     clDodgerBlue32', $FF1E90FF ));
19309:     clFireBrick32', $FFB22222 ));
19310:     clFloralWhite32', $FFFFFAF0 ));
19311:     clGainsBoro32', $FFDCDCDC ));
19312:     clGhostWhite32', $FFF8F8FF ));
19313:     clGold32', $FFFFD700 ));
19314:     clGoldenRod32', $FFDAA520 ));
19315:     clGreenYellow32', $FFADFF2F ));
19316:     clGrey32', $FF808080 ));
19317:     clHoneyDew32', $FFF0FFF0 ));
19318:     clHotPink32', $FFFF69B4 ));
19319:     clIndianRed32', $FFCD5C5C ));
19320:     clIndigo32', $FF4B0082 ));
19321:     clIvory32', $FFFFFFF0 ));
19322:     clKhaki32', $FFF0E68C ));
19323:     clLavender32', $FFE6E6FA ));
19324:     clLavenderBlush32', $FFFFF0F5 ));
19325:     clLawnGreen32', $FF7CFC00 ));
19326:     clLemonChiffon32', $FFFFFACD ));
19327:     clLightBlue32', $FFADD8E6 ));
19328:     clLightCoral32', $FFF08080 ));
19329:     clLightCyan32', $FFE0FFFF ));
19330:     clLightGoldenRodYellow32', $FFFAFAD2 ));
19331:     clLightGreen32', $FF90EE90 ));
19332:     clLightGrey32', $FFD3D3D3 ));
19333:     clLightPink32', $FFFFB6C1 ));
19334:     clLightSalmon32', $FFFFA07A ));
19335:     clLightSeagreen32', $FF20B2AA ));
19336:     clLightSkyblue32', $FF87CEFA ));
19337:     clLightSlategray32', $FF778899 ));
19338:     clLightSlategrey32', $FF778899 ));
19339:     clLightSteelblue32', $FFB0C4DE ));
19340:     clLightYellow32', $FFFFFFE0 ));
19341:     clLtGray32', $FFC0C0C0 ));
19342:     clMedGray32', $FFA0A0A4 ));
19343:     clDkGray32', $FF808080 ));
19344:     clMoneyGreen32', $FFC0DCC0 ));
19345:     clLegacySkyBlue32', $FFA6CAF0 ));
19346:     clCream32', $FFFFFBF0 ));
19347:     clLimeGreen32', $FF32CD32 ));
19348:     clLinen32', $FFFAF0E6 ));
19349:     clMediumAquamarine32', $FF66CDAA ));
19350:     clMediumBlue32', $FF0000CD ));
19351:     clMediumOrchid32', $FFBA55D3 ));
19352:     clMediumPurple32', $FF9370DB ));
19353:     clMediumSeaGreen32', $FF3CB371 ));
19354:     clMediumSlateBlue32', $FF7B68EE ));
19355:     clMediumSpringGreen32', $FF00FA9A ));
19356:     clMediumTurquoise32', $FF48D1CC ));
19357:     clMediumVioletRed32', $FFC71585 ));
19358:     clMidnightBlue32', $FF191970 ));
19359:     clMintCream32', $FFF5FFFA ));
19360:     clMistyRose32', $FFFFE4E1 ));
19361:     clMoccasin32', $FFFFE4B5 ));
19362:     clNavajoWhite32', $FFFFDEAD ));
19363:     clOldLace32', $FFFDF5E6 ));
19364:     clOliveDrab32', $FF6B8E23 ));
19365:     clOrange32', $FFFFA500 ));
19366:     clOrangeRed32', $FFFF4500 ));
19367:     clOrchid32', $FFDA70D6 ));
19368:     clPaleGoldenRod32', $FFEEE8AA ));
19369:     clPaleGreen32', $FF98FB98 ));
19370:     clPaleTurquoise32', $FFAFEEEE ));
19371:     clPaleVioletred32', $FFDB7093 ));
19372:     clPapayaWhip32', $FFFFEFD5 ));
19373:     clPeachPuff32', $FFFFDAB9 ));
```

```
19374:     clPeru32', $FFCD853F ));
19375:     clPlum32', $FFDDA0DD ));
19376:     clPowderBlue32', $FFB0E0E6 ));
19377:     clRosyBrown32', $FFBC8F8F ));
19378:     clRoyalBlue32', $FF4169E1 ));
19379:     clSaddleBrown32', $FF8B4513 ));
19380:     clSalmon32', $FFFA8072 ));
19381:     clSandyBrown32', $FFF4A460 ));
19382:     clSeaGreen32', $FF2E8B57 ));
19383:     clSeaShell32', $FFFFF5EE ));
19384:     clSienna32', $FFA0522D ));
19385:     clSilver32', $FFC0C0C0 ));
19386:     clSkyblue32', $FF87CEEB ));
19387:     clSlateBlue32', $FF6A5ACD ));
19388:     clSlateGray32', $FF708090 ));
19389:     clSlateGrey32', $FF708090 ));
19390:     clSnow32', $FFFFFAFA ));
19391:     clSpringgreen32', $FF00FF7F ));
19392:     clSteelblue32', $FF4682B4 ));
19393:     clTan32', $FFD2B48C ));
19394:     clThistle32', $FFD8BFD8 ));
19395:     clTomato32', $FFFF6347 ));
19396:     clTurquoise32', $FF40E0D0 ));
19397:     clViolet32', $FFEE82EE ));
19398:     clWheat32', $FFF5DEB3 ));
19399:     clWhitesmoke32', $FFF5F5F5 ));
19400:     clYellowgreen32', $FF9ACD32 ));
19401:     clTrWhite32', $7FFFFFFF ));
19402:     clTrBlack32', $7F000000 ));
19403:     clTrRed32', $7FFF0000 ));
19404:     clTrGreen32', $7F00FF00 ));
19405:     clTrBlue32', $7F0000FF ));
19406:   // Fixed point math constants
19407:   FixedOne = $10000;  FixedHalf = $7FFF;
19408:   FixedPI  = Round(PI * FixedOne);
19409:   FixedToFloat = 1/FixedOne;
19410:
19411:  Special Types
19412: ****************************************************
19413:   type Complex = record
19414:     X, Y : Float;
19415:   end;
19416:   type TVector     = array of Float;
19417:   TIntVector  = array of Integer;
19418:   TCompVector = array of Complex;
19419:   TBoolVector = array of Boolean;
19420:   TStrVector  = array of String;
19421:   TMatrix     = array of TVector;
19422:   TIntMatrix  = array of TIntVector;
19423:   TCompMatrix = array of TCompVector;
19424:   TBoolMatrix = array of TBoolVector;
19425:   TStrMatrix  = array of TStrVector;
19426:   TByteArray = array[0..32767] of byte; !
19427:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
19428:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
19429:   T2StringArray = array of array of string;
19430:   T2IntegerArray = array of array of integer;
19431:   AddTypeS('INT_PTR', 'Integer
19432:   AddTypeS('LONG_PTR', 'Integer
19433:   AddTypeS('UINT_PTR', 'Cardinal
19434:   AddTypeS('ULONG_PTR', 'Cardinal
19435:   AddTypeS('DWORD_PTR', 'ULONG_PTR
19436:   TIntegerDynArray', 'array of Integer
19437:   TCardinalDynArray', 'array of Cardinal
19438:   TWordDynArray', 'array of Word
19439:   TSmallIntDynArray', 'array of SmallInt
19440:   TByteDynArray', 'array of Byte
19441:   TShortIntDynArray', 'array of ShortInt
19442:   TInt64DynArray', 'array of Int64
19443:   TLongWordDynArray', 'array of LongWord
19444:   TSingleDynArray', 'array of Single
19445:   TDoubleDynArray', 'array of Double
19446:   TBooleanDynArray', 'array of Boolean
19447:   TStringDynArray', 'array of string
19448:   TWideStringDynArray', 'array of WideString
19449:   TDynByteArray     = array of Byte;
19450:   TDynShortintArray = array of Shortint;
19451:   TDynSmallintArray = array of Smallint;
19452:   TDynWordArray     = array of Word;
19453:   TDynIntegerArray  = array of Integer;
19454:   TDynLongintArray  = array of Longint;
19455:   TDynCardinalArray = array of Cardinal;
19456:   TDynInt64Array    = array of Int64;
19457:   TDynExtendedArray = array of Extended;
19458:   TDynDoubleArray   = array of Double;
19459:   TDynSingleArray   = array of Single;
19460:   TDynFloatArray    = array of Float;
19461:   TDynPointerArray  = array of Pointer;
19462:   TDynStringArray   = array of string;
```

```
19463:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
19464:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
19465:   TSynSearchOptions = set of TSynSearchOption;
19466:
19467:
19468: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
19469: ------------------------------------------------------------------------------
19470: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
19471: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
19472: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
19473: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19474: function CheckStringSum(vstring: string): integer;
19475: function HexToInt(HexNum: string): LongInt;
19476: function IntToBin(Int: Integer): String;
19477: function BinToInt(Binary: String): Integer;
19478: function HexToBin(HexNum: string): string; external2
19479: function BinToHex(Binary: String): string;
19480: function IntToFloat(i: Integer): double;
19481: function AddThousandSeparator(S: string; myChr: Char): string;
19482: function Max3(const X,Y,Z: Integer): Integer;
19483: procedure Swap(var X,Y: char); // faster without inline
19484: procedure ReverseString(var S: String);
19485: function CharToHexStr(Value: Char): string;
19486: function CharToUniCode(Value: Char): string;
19487: function Hex2Dec(Value: Str002): Byte;
19488: function HexStrCodeToStr(Value: string): string;
19489: function HexToStr(i: integer; value: string): string;
19490: function UniCodeToStr(Value: string): string;
19491: function CRC16(statement: string): string;
19492: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
19493: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
19494: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19495: Procedure ExecuteCommand(executeFile, paramstring: string);
19496: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
19497: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
19498: procedure SearchAndOpenDoc(vfilenamepath: string);
19499: procedure ShowInterfaces(myFile: string);
19500: function Fact2(av: integer): extended;
19501: Function BoolToStr(B: Boolean): string;
19502: Function GCD(x, y : LongInt) : LongInt;
19503: function LCM(m,n: longint): longint;
19504: function GetASCII: string;
19505: function GetItemHeight(Font: TFont): Integer;
19506: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
19507: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
19508: function getHINSTANCE: longword;
19509: function getHMODULE: longword;
19510: function GetASCII: string;
19511: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
19512: function WordIsOk(const AWord: string; var VW: Word): boolean;
19513: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
19514: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
19515: function SafeStr(const s: string): string;
19516: function ExtractUrlPath(const FileName: string): string;
19517: function ExtractUrlName(const FileName: string): string;
19518: function IsInternet: boolean;
19519: function RotateLeft1Bit_u32( Value: uint32): uint32;
19520: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats : Boolean);
19521: procedure getEnvironmentInfo;
19522: procedure AntiFreeze;
19523: function GetCPUSpeed: Double;
19524: function IsVirtualPcGuest : Boolean;
19525: function IsVmWareGuest : Boolean;
19526: procedure StartSerialDialog;
19527: function IsWoW64: boolean;
19528: function IsWow64String(var s: string): Boolean;
19529: procedure StartThreadDemo;
19530: Function RGB(R,G,B: Byte): TColor;
19531: Function Sendln(amess: string): boolean;
19532: Procedure maXbox;
19533: Function AspectRatio(aWidth, aHeight: Integer): String;
19534: function wget(aURL, afile: string): boolean;
19535: procedure PrintList(Value: TStringList);
19536: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
19537: procedure getEnvironmentInfo;
19538: procedure AntiFreeze;
19539: function getBitmap(apath: string): TBitmap;
19540: procedure ShowMessageBig(const aText : string);
19541: function YesNoDialog(const ACaption, AMsg: string): boolean;
19542: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
19543: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19544: //function myStrToBytes(const Value: String): TBytes;
19545: //function myBytesToStr(const Value: TBytes): String;
19546: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19547: function getBitmap(apath: string): TBitmap;
19548: procedure ShowMessageBig(const aText : string);
19549: Function StrToBytes(const Value: String): TBytes;
19550: Function BytesToStr(const Value: TBytes): String;
```

```
19551: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19552: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
       HostName:String):Bool;
19553: function FindInPaths(const fileName, paths : String) : String;
19554: procedure initHexArray(var hexn: THexArray);
19555: function josephusG(n,k: integer; var graphout: string): integer;
19556: function isPowerof2(num: int64): boolean;
19557: function powerOf2(exponent: integer): int64;
19558: function getBigPI: string;
19559: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
19560:  function GetASCIILine: string;
19561: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
19562:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
19563: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
19564: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
19565: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
19566: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
19567: function isKeypressed: boolean;
19568: function Keypress: boolean;
19569: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19570: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
19571: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
19572: function GetOSName: string;
19573: function GetOSVersion: string;
19574: function GetOSNumber: string;
19575: function getEnvironmentString: string;
19576: procedure StrReplace(var Str: String; Old, New: String);
19577: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19578: function getTeamViewerID: string;
19579: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
19580: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
19581: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19582: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
19583: function StartSocketService: Boolean;
19584: procedure StartSocketServiceForm;
19585: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
19586: function GetFileList1(apath: string): TStringlist;
19587: procedure LetFileList(FileList: TStringlist; apath: string);
19588: procedure StartWeb(aurl: string);
19589: function GetTodayFiles(startdir, amask: string): TStringlist;
19590: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
19591: function JavahashCode(val: string): Integer;
19592: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
19593: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
19594: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
19595: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
19596: Procedure ConvertToGray(Cnv: TCanvas);
19597: function GetFileDate(aFile:string; aWithTime:Boolean):string;
19598: procedure ShowMemory;
19599: function ShowMemory2: string;
19600: function getHostIP: string;
19601: procedure ShowBitmap(bmap: TBitmap);
19602: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
19603: function CreateDBGridForm(dblist: TStringList): TListbox;
19604: function isService: boolean;
19605: function isApplication: boolean;
19606: function isTerminalSession: boolean;
19607:
19608:
19609: // News of 3.9.8 up
19610: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19611: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19612: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19613: JvChart - TJvChart Component - 2009 Public
19614: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
19615: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
19616: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
19617: DMath DLL included incl. Demos
19618: Interface Navigator menu/View/Intf Navigator
19619: Unit Explorer menu/Debug/Units Explorer
19620: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
19621: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
19622: Script History to 9 Files WebServer light /Options/Addons/WebServer
19623: Full Text Finder, JVSimLogic Simulator Package
19624: Halt-Stop Program in Menu, WebServer2, Stop Event ,
19625: Conversion Routines, Prebuild Forms, CodeSearch
19626: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19627: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19628: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19629: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
19630: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
19631: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
19632: IDE Reflection API, Session Service Shell S3
19633: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
19634: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
19635: arduino map() function, PMRandom Generator
19636: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
19637: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
19638: REST Test Lib, Multilang Component, Forth Interpreter
```

```
19639: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
19640: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
19641: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19642: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19643: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
19644: QRCode Service, add more CFunctions like CDateTime of Synapse
19645: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
19646: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
19647: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
19648: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
19649: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
19650: BOLD Package, Indy Package5, maTRIx. MATHEMAX
19651: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
19652: emax layers: system-package-component-unit-class-function-block
19653: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
19654: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
19655: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
19656: OpenGL Game Demo: ..Options/Add Ons/Reversi
19657: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
19658: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
19659: 7% performance gain (hot spot profiling)
19660: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
19661: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
19662: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
19663: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
19664:
19665: add routines in 3.9.7.5
19666: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
19667: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
19668: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
19669: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
19670: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
19671: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
19672: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
19673:
19674: ////////////////////////// TestUnits ////////////////////////////
19675:   SelftestPEM;
19676:   SelfTestCFundamentUtils;
19677:   SelfTestCFileUtils;
19678:   SelfTestCDateTime;
19679:   SelfTestCTimer;
19680:   SelfTestCRandom;
19681:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
19682:            Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
19683:
19684:   // Note: There's no need for installing a client certificate in the
19685:   //       webbrowser. The server asks the webbrowser to send a certificate but
19686:   //       if nothing is installed the software will work because the server
19687:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
19688:   //       file: c_cacert.p12  password: c_cakey
19689:
19690:   TGraphicControl = class(TControl)
19691:   private
19692:     FCanvas: TCanvas;
19693:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19694:   protected
19695:     procedure Paint; virtual;
19696:     property Canvas: TCanvas read FCanvas;
19697:   public
19698:     constructor Create(AOwner: TComponent); override;
19699:     destructor Destroy; override;
19700:   end;
19701:
19702:   TCustomControl = class(TWinControl)
19703:   private
19704:     FCanvas: TCanvas;
19705:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19706:   protected
19707:     procedure Paint; virtual;
19708:     procedure PaintWindow(DC: HDC); override;
19709:     property Canvas: TCanvas read FCanvas;
19710:   public
19711:     constructor Create(AOwner: TComponent); override;
19712:     destructor Destroy; override;
19713:   end;
19714:     RegisterPublishedProperties;
19715:     ('ONCHANGE', 'TNotifyEvent', iptrw);
19716:     ('ONCLICK', 'TNotifyEvent', iptrw);
19717:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
19718:     ('ONENTER', 'TNotifyEvent', iptrw);
19719:     ('ONEXIT', 'TNotifyEvent', iptrw);
19720:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
19721:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
19722:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
19723:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
19724:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
19725: //************************************************************************
19726:   // To stop the while loop, click on Options/Show Include (boolean switch)!
19727:   Control a loop in a script with a form event:
```

```
19728:   IncludeON;   //control the while loop
19729:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
19730:   repeat {for it:= 1 to n do} until is keypressed  //keypress in output window below (memo2)
19731:
19732: //-------------------------------------------------------------------------
19733: //*************mX4 ini-file Configuration*********************************
19734: //-------------------------------------------------------------------------
19735:   using config file maxboxdef.ini       menu/Help/Config File
19736:
19737: //*** Definitions for maXbox mX3 ***
19738: [FORM]
19739: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
19740: FONTSIZE=14
19741: EXTENSION=txt
19742: SCREENX=1386
19743: SCREENY=1077
19744: MEMHEIGHT=350
19745: PRINTFONT=Courier New //GUI Settings
19746: LINENUMBERS=Y   //line numbers at gutter in editor at left side
19747: EXCEPTIONLOG=Y  //store excepts + success in 2 log files see below! –menu Debug/Show Last Exceptions
19748: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
19749: BOOTSCRIPT=Y    //enabling load a boot script
19750: MEMORYREPORT=Y  //shows memory report on closing maXbox
19751: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
19752: NAVIGATOR=N     //shows function list at the right side of editor
19753: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
19754: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
19755: [WEB]
19756: IPPORT=8080     //for internal webserver – menu /Options/Add Ons/WebServer
19757: IPHOST=192.168.1.53
19758: ROOTCERT=filepathY
19759: SCERT=filepathY
19760: RSAKEY=filepathY
19761: VERSIONCHECK=Y
19762:
19763: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
19764:
19765: Also possible to set report memory in script to override ini setting
19766: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19767:
19768:
19769: After Change the ini file you can reload the file with ../Help/Config Update
19770:
19771: //-------------------------------------------------------------------------
19772: //*************mX4 maildef.ini ini-file Configuration*********************
19773: //-------------------------------------------------------------------------
19774: //*** Definitions for maXMail ***
19775: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
19776: [MAXMAIL]
19777: HOST=getmail.softwareschule.ch
19778: USER=mailusername
19779: PASS=password
19780: PORT=110
19781: SSL=Y
19782: BODY=Y
19783: LAST=5
19784:
19785: ADO Connection String:
19786: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
19787:
19788: \452_dbtreeview2access.txt
19789: program TestDbTreeViewMainForm2_ACCESS;
19790:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
19791:                    +Exepath+'\examples\detail.mdb;Persist Security Info=False';
19792:
19793: OpenSSL Lib:  unit ssl_openssl_lib;
19794: {$IFDEF CIL}
19795: const
19796:   {$IFDEF LINUX}
19797:   DLLSSLName = 'libssl.so';
19798:   DLLUtilName = 'libcrypto.so';
19799:   {$ELSE}
19800:   DLLSSLName = 'ssleay32.dll';
19801:   DLLUtilName = 'libeay32.dll';
19802:   {$ENDIF}
19803: {$ELSE}
19804: var
19805:   {$IFNDEF MSWINDOWS}
19806:     {$IFDEF DARWIN}
19807:     DLLSSLName: string = 'libssl.dylib';
19808:     DLLUtilName: string = 'libcrypto.dylib';
19809:     {$ELSE}
19810:     DLLSSLName: string = 'libssl.so';
19811:     DLLUtilName: string = 'libcrypto.so';
19812:     {$ENDIF}
19813:   {$ELSE}
19814:   DLLSSLName: string = 'ssleay32.dll';
19815:   DLLSSLName2: string = 'libssl32.dll';
19816:   DLLUtilName: string = 'libeay32.dll';
```

```
19817:   {$ENDIF}
19818: {$ENDIF}
19819:
19820:
19821: //------------------------------------------------------------------------------
19822: //**************mX4 Macro Tags  ***********************************************
19823: //------------------------------------------------------------ ----------------------
19824:
19825:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
       E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
19826:
19827: //Tag Macros
19828:
19829:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
19830:
19831: //Tag Macros
19832: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
19833: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
19834: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
19835: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
19836: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
19837: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
19838: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
19839: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
19840: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
19841:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
19842: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
19843: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
19844:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
19845: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
19846:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
19847: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
19848:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
19849:
19850: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
19851:
19852: //Replace Macros
19853:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
19854:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
19855:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
19856:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
19857:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
19858:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
19859:
19860:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
19861:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
19862: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
19863:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
19864:
19865: //------------------------------------------------------------------------------
19866: //**************mX4 ToDo List Tags  ../Help/ToDo List************************
19867: //------------------------------------------------------------ ----------------------
19868:
19869:     while I < sl.Count do begin
19870: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
19871:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
19872:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
19873:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
19874:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
19875:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
19876:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
19877:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
19878:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
19879:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
19880:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
19881:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
19882:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
19883:       Inc(I);
19884:     end;
19885:
19886:
19887: //------------------------------------------------------------------------------
19888: //**************mX4 Public  Tools API ****************************************
19889: //------------------------------------------------------------------------------
19890:   file : unit uPSI_fMain.pas;               {$OTAP} Open Tools API Catalog
19891:  // Those functions concern the editor and preprocessor, all of the IDE
19892:  Example: Call it with maxform1.Info1Click(self)
19893:  Note: Call all Methods with maxForm1., e.g.:
19894:                        maxForm1.ShellStyle1Click(self);
19895:
19896: procedure SIRegister_fMain(CL: TPSPascalCompiler);
19897: begin
19898:  Const('BYTECODE','String 'bytecode.txt'
19899:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
19900:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
19901:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
19902:  Const('PSINC','String PS Includes (*.inc)|*.INC
19903:  Const('DEFFILENAME','String 'firstdemo.txt
19904:  Const('DEFINIFILE','String 'maxboxdef.ini
```

```
19905:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
19906:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
19907:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
19908:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
19909:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
19910:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
19911:  Const('INCLUDEBOX','String 'pas_includebox.inc
19912:  Const('BOOTSCRIPT','String 'maxbootscript.txt
19913:  Const('MBVERSION','String '3.9.9.98
19914:  Const('VERSION','String'3.9.9.98
19915:  Const('MBVER','String '399
19916:  Const('MBVERI','Integer'(399);
19917:  Const('MBVERIALL','Integer'(39998);
19918:  Const('EXENAME','String 'maXbox3.exe
19919:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
19920:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
19921:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
19922:  Const('MXINTERNETCHECK','String 'www.ask.com
19923:  Const('MXMAIL','String 'max@kleiner.com
19924:  Const('TAB','Char #$09);
19925:  Const('CODECOMPLETION','String 'bds_delphi.dci
19926:  SIRegister_TMaxForm1(CL);
19927: end;
19928:
19929:   with FindClass('TForm'),'TMaxForm1') do begin
19930:     memo2', 'TMemo', iptrw);
19931:     memo1', 'TSynMemo', iptrw);
19932:     CB1SCList', 'TComboBox', iptrw);
19933:     mxNavigator', 'TComboBox', iptrw);
19934:     IPHost', 'string', iptrw);
19935:     IPPort', 'integer', iptrw);
19936:     COMPort', 'integer', iptrw);      //3.9.6.4
19937:     Splitter1', 'TSplitter', iptrw);
19938:     PSScript', 'TPSScript', iptrw);
19939:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
19940:     MainMenu1', 'TMainMenu', iptrw);
19941:     Program1', 'TMenuItem', iptrw);
19942:     Compile1', 'TMenuItem', iptrw);
19943:     Files1', 'TMenuItem', iptrw);
19944:     open1', 'TMenuItem', iptrw);
19945:     Save2', 'TMenuItem', iptrw);
19946:     Options1', 'TMenuItem', iptrw);
19947:     Savebefore1', 'TMenuItem', iptrw);
19948:     Largefont1', 'TMenuItem', iptrw);
19949:     sBytecode1', 'TMenuItem', iptrw);
19950:     Saveas3', 'TMenuItem', iptrw);
19951:     Clear1', 'TMenuItem', iptrw);
19952:     Slinenumbers1', 'TMenuItem', iptrw);
19953:     About1', 'TMenuItem', iptrw);
19954:     Search1', 'TMenuItem', iptrw);
19955:     SynPasSyn1', 'TSynPasSyn', iptrw);
19956:     memo1', 'TSynMemo', iptrw);
19957:     SynEditSearch1', 'TSynEditSearch', iptrw);
19958:     WordWrap1', 'TMenuItem', iptrw);
19959:     XPManifest1', 'TXPManifest', iptrw);
19960:     SearchNext1', 'TMenuItem', iptrw);
19961:     Replace1', 'TMenuItem', iptrw);
19962:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
19963:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
19964:     ShowInclude1', 'TMenuItem', iptrw);
19965:     SynEditPrint1', 'TSynEditPrint', iptrw);
19966:     Printout1', 'TMenuItem', iptrw);
19967:     mnPrintColors1', 'TMenuItem', iptrw);
19968:     dlgFilePrint', 'TPrintDialog', iptrw);
19969:     dlgPrintFont1', 'TFontDialog', iptrw);
19970:     mnuPrintFont1', 'TMenuItem', iptrw);
19971:     Include1', 'TMenuItem', iptrw);
19972:     CodeCompletionList1', 'TMenuItem', iptrw);
19973:     IncludeList1', 'TMenuItem', iptrw);
19974:     ImageList1', 'TImageList', iptrw);
19975:     ImageList2', 'TImageList', iptrw);
19976:     CoolBar1', 'TCoolBar', iptrw);
19977:     ToolBar1', 'TToolBar', iptrw);
19978:     tbtnLoad', 'TToolButton', iptrw);
19979:     ToolButton2', 'TToolButton', iptrw);
19980:     tbtnFind', 'TToolButton', iptrw);
19981:     tbtnCompile', 'TToolButton', iptrw);
19982:     tbtnTrans', 'TToolButton', iptrw);
19983:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
19984:     toolbtnTutorial', 'TToolButton', iptrw);
19985:     tbtn6res', 'TToolButton', iptrw);
19986:     ToolButton5', 'TToolButton', iptrw);
19987:     ToolButton1', 'TToolButton', iptrw);
19988:     ToolButton3', 'TToolButton', iptrw);
19989:     statusBar1', 'TStatusBar', iptrw);
19990:     SaveOutput1', 'TMenuItem', iptrw);
19991:     ExportClipboard1', 'TMenuItem', iptrw);
19992:     Close1', 'TMenuItem', iptrw);
19993:     Manual1', 'TMenuItem', iptrw);
```

```
19994:     About2', 'TMenuItem', iptrw);
19995:     loadLastfile1', 'TMenuItem', iptrw);
19996:     imglogo', 'TImage', iptrw);
19997:     cedebug', 'TPSScriptDebugger', iptrw);
19998:     debugPopupMenu1', 'TPopupMenu', iptrw);
19999:     BreakPointMenu', 'TMenuItem', iptrw);
20000:     Decompile1', 'TMenuItem', iptrw);
20001:     StepInto1', 'TMenuItem', iptrw);
20002:     StepOut1', 'TMenuItem', iptrw);
20003:     Reset1', 'TMenuItem', iptrw);
20004:     DebugRun1', 'TMenuItem', iptrw);
20005:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
20006:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
20007:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
20008:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
20009:     tutorial4', 'TMenuItem', iptrw);
20010:     ExporttoClipboard1', 'TMenuItem', iptrw);
20011:     ImportfromClipboard1', 'TMenuItem', iptrw);
20012:     N4', 'TMenuItem', iptrw);
20013:     N5', 'TMenuItem', iptrw);
20014:     N6', 'TMenuItem', iptrw);
20015:     ImportfromClipboard2', 'TMenuItem', iptrw);
20016:     tutorial1', 'TMenuItem', iptrw);
20017:     N7', 'TMenuItem', iptrw);
20018:     ShowSpecChars1', 'TMenuItem', iptrw);
20019:     OpenDirectory1', 'TMenuItem', iptrw);
20020:     procMess', 'TMenuItem', iptrw);
20021:     tbtnUseCase', 'TToolButton', iptrw);
20022:     ToolButton7', 'TToolButton', iptrw);
20023:     EditFont1', 'TMenuItem', iptrw);
20024:     UseCase1', 'TMenuItem', iptrw);
20025:     tutorial21', 'TMenuItem', iptrw);
20026:     OpenUseCase1', 'TMenuItem', iptrw);
20027:     PSImport_DB1', 'TPSImport_DB', iptrw);
20028:     tutorial31', 'TMenuItem', iptrw);
20029:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
20030:     HTMLSyntax1', 'TMenuItem', iptrw);
20031:     ShowInterfaces1', 'TMenuItem', iptrw);
20032:     Tutorial5', 'TMenuItem', iptrw);
20033:     AllFunctionsList1', 'TMenuItem', iptrw);
20034:     ShowLastException1', 'TMenuItem', iptrw);
20035:     PlayMP31', 'TMenuItem', iptrw);
20036:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
20037:     texSyntax1', 'TMenuItem', iptrw);
20038:     N8', 'TMenuItem', iptrw);
20039:     GetEMails1', 'TMenuItem', iptrw);
20040:     SynCppSyn1', 'TSynCppSyn', iptrw);
20041:     CSyntax1', 'TMenuItem', iptrw);
20042:     Tutorial6', 'TMenuItem', iptrw);
20043:     New1', 'TMenuItem', iptrw);
20044:     AllObjectsList1', 'TMenuItem', iptrw);
20045:     LoadBytecode1', 'TMenuItem', iptrw);
20046:     CipherFile1', 'TMenuItem', iptrw);
20047:     N9', 'TMenuItem', iptrw);
20048:     N10', 'TMenuItem', iptrw);
20049:     Tutorial11', 'TMenuItem', iptrw);
20050:     Tutorial71', 'TMenuItem', iptrw);
20051:     UpdateService1', 'TMenuItem', iptrw);
20052:     PascalSchool1', 'TMenuItem', iptrw);
20053:     Tutorial81', 'TMenuItem', iptrw);
20054:     DelphiSite1', 'TMenuItem', iptrw);
20055:     Output1', 'TMenuItem', iptrw);
20056:     TerminalStyle1', 'TMenuItem', iptrw);
20057:     ReadOnly1', 'TMenuItem', iptrw);
20058:     ShellStyle1', 'TMenuItem', iptrw);
20059:     BigScreen1', 'TMenuItem', iptrw);
20060:     Tutorial91', 'TMenuItem', iptrw);
20061:     SaveOutput2', 'TMenuItem', iptrw);
20062:     N11', 'TMenuItem', iptrw);
20063:     SaveScreenshot', 'TMenuItem', iptrw);
20064:     Tutorial101', 'TMenuItem', iptrw);
20065:     SQLSyntax1', 'TMenuItem', iptrw);
20066:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
20067:     Console1', 'TMenuItem', iptrw);
20068:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
20069:     XMLSyntax1', 'TMenuItem', iptrw);
20070:     ComponentCount1', 'TMenuItem', iptrw);
20071:     NewInstance1', 'TMenuItem', iptrw);
20072:     toolbtnTutorial', 'TToolButton', iptrw);
20073:     Memory1', 'TMenuItem', iptrw);
20074:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
20075:     JavaSyntax1', 'TMenuItem', iptrw);
20076:     SyntaxCheck1', 'TMenuItem', iptrw);
20077:     Tutorial10Statistics1', 'TMenuItem', iptrw);
20078:     ScriptExplorer1', 'TMenuItem', iptrw);
20079:     FormOutput1', 'TMenuItem', iptrw);
20080:     ArduinoDump1', 'TMenuItem', iptrw);
20081:     AndroidDump1', 'TMenuItem', iptrw);
20082:     GotoEnd1', 'TMenuItem', iptrw);
```

```
20083:     AllResourceList1', 'TMenuItem', iptrw);
20084:     ToolButton4', 'TToolButton', iptrw);
20085:     tbtn6res', 'TToolButton', iptrw);
20086:     Tutorial11Forms1', 'TMenuItem', iptrw);
20087:     Tutorial12SQL1', 'TMenuItem', iptrw);
20088:     ResourceExplore1', 'TMenuItem', iptrw);
20089:     Info1', 'TMenuItem', iptrw);
20090:     N12', 'TMenuItem', iptrw);
20091:     CryptoBox1', 'TMenuItem', iptrw);
20092:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
20093:     CipherFile2', 'TMenuItem', iptrw);
20094:     N13', 'TMenuItem', iptrw);
20095:     ModulesCount1', 'TMenuItem', iptrw);
20096:     AddOns2', 'TMenuItem', iptrw);
20097:     N4GewinntGame1', 'TMenuItem', iptrw);
20098:     DocuforAddOns1', 'TMenuItem', iptrw);
20099:     Tutorial14Async1', 'TMenuItem', iptrw);
20100:     Lessons15Review1', 'TMenuItem', iptrw);
20101:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
20102:     PHPSyntax1', 'TMenuItem', iptrw);
20103:     Breakpoint1', 'TMenuItem', iptrw);
20104:     SerialRS2321', 'TMenuItem', iptrw);
20105:     N14', 'TMenuItem', iptrw);
20106:     SynCSSyn1', 'TSynCSSyn', iptrw);
20107:     CSyntax2', 'TMenuItem', iptrw);
20108:     Calculator1', 'TMenuItem', iptrw);
20109:     tbtnSerial', 'TToolButton', iptrw);
20110:     ToolButton8', 'TToolButton', iptrw);
20111:     Tutorial151', 'TMenuItem', iptrw);
20112:     N15', 'TMenuItem', iptrw);
20113:     N16', 'TMenuItem', iptrw);
20114:     ControlBar1', 'TControlBar', iptrw);
20115:     ToolBar2', 'TToolBar', iptrw);
20116:     BtnOpen', 'TToolButton', iptrw);
20117:     BtnSave', 'TToolButton', iptrw);
20118:     BtnPrint', 'TToolButton', iptrw);
20119:     BtnColors', 'TToolButton', iptrw);
20120:     btnClassReport', 'TToolButton', iptrw);
20121:     BtnRotateRight', 'TToolButton', iptrw);
20122:     BtnFullSize', 'TToolButton', iptrw);
20123:     BtnFitToWindowSize', 'TToolButton', iptrw);
20124:     BtnZoomMinus', 'TToolButton', iptrw);
20125:     BtnZoomPlus', 'TToolButton', iptrw);
20126:     Panel1', 'TPanel', iptrw);
20127:     LabelBrettgroesse', 'TLabel', iptrw);
20128:     CB1SCList', 'TComboBox', iptrw);
20129:     ImageListNormal', 'TImageList', iptrw);
20130:     spbtnexplore', 'TSpeedButton', iptrw);
20131:     spbtnexample', 'TSpeedButton', iptrw);
20132:     spbsaveas', 'TSpeedButton', iptrw);
20133:     imglogobox', 'TImage', iptrw);
20134:     EnlargeFont1', 'TMenuItem', iptrw);
20135:     EnlargeFont2', 'TMenuItem', iptrw);
20136:     ShrinkFont1', 'TMenuItem', iptrw);
20137:     ThreadDemo1', 'TMenuItem', iptrw);
20138:     HEXEditor1', 'TMenuItem', iptrw);
20139:     HEXView1', 'TMenuItem', iptrw);
20140:     HEXInspect1', 'TMenuItem', iptrw);
20141:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
20142:     ExporttoHTML1', 'TMenuItem', iptrw);
20143:     ClassCount1', 'TMenuItem', iptrw);
20144:     HTMLOutput1', 'TMenuItem', iptrw);
20145:     HEXEditor2', 'TMenuItem', iptrw);
20146:     Minesweeper1', 'TMenuItem', iptrw);
20147:     N17', 'TMenuItem', iptrw);
20148:     PicturePuzzle1', 'TMenuItem', iptrw);
20149:     sbvclhelp', 'TSpeedButton', iptrw);
20150:     DependencyWalker1', 'TMenuItem', iptrw);
20151:     WebScanner1', 'TMenuItem', iptrw);
20152:     View1', 'TMenuItem', iptrw);
20153:     mnToolbar1', 'TMenuItem', iptrw);
20154:     mnStatusbar2', 'TMenuItem', iptrw);
20155:     mnConsole2', 'TMenuItem', iptrw);
20156:     mnCoolbar2', 'TMenuItem', iptrw);
20157:     mnSplitter2', 'TMenuItem', iptrw);
20158:     WebServer1', 'TMenuItem', iptrw);
20159:     Tutorial17Server1', 'TMenuItem', iptrw);
20160:     Tutorial18Arduino1', 'TMenuItem', iptrw);
20161:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
20162:     PerlSyntax1', 'TMenuItem', iptrw);
20163:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
20164:     PythonSyntax1', 'TMenuItem', iptrw);
20165:     DMathLibrary1', 'TMenuItem', iptrw);
20166:     IntfNavigator1', 'TMenuItem', iptrw);
20167:     EnlargeFontConsole1', 'TMenuItem', iptrw);
20168:     ShrinkFontConsole1', 'TMenuItem', iptrw);
20169:     SetInterfaceList1', 'TMenuItem', iptrw);
20170:     popintfList', 'TPopupMenu', iptrw);
20171:     intfAdd1', 'TMenuItem', iptrw);
```

```
20172:     intfDelete1', 'TMenuItem', iptrw);
20173:     intfRefactor1', 'TMenuItem', iptrw);
20174:     Defactor1', 'TMenuItem', iptrw);
20175:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
20176:     Tutorial20Regex', 'TMenuItem', iptrw);
20177:     N18', 'TMenuItem', iptrw);
20178:     ManualE1', 'TMenuItem', iptrw);
20179:     FullTextFinder1', 'TMenuItem', iptrw);
20180:     Move1', 'TMenuItem', iptrw);
20181:     FractalDemo1', 'TMenuItem', iptrw);
20182:     Tutorial21Android1', 'TMenuItem', iptrw);
20183:     Tutorial0Function1', 'TMenuItem', iptrw);
20184:     SimuLogBox1', 'TMenuItem', iptrw);
20185:     OpenExamples1', 'TMenuItem', iptrw);
20186:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
20187:     JavaScriptSyntax1', 'TMenuItem', iptrw);
20188:     Halt1', 'TMenuItem', iptrw);
20189:     CodeSearch1', 'TMenuItem', iptrw);
20190:     SynRubySyn1', 'TSynRubySyn', iptrw);
20191:     RubySyntax1', 'TMenuItem', iptrw);
20192:     Undo1', 'TMenuItem', iptrw);
20193:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
20194:     LinuxShellScript1', 'TMenuItem', iptrw);
20195:     Rename1', 'TMenuItem', iptrw);
20196:     spdcodesearch', 'TSpeedButton', iptrw);
20197:     Preview1', 'TMenuItem', iptrw);
20198:     Tutorial22Services1', 'TMenuItem', iptrw);
20199:     Tutorial23RealTime1', 'TMenuItem', iptrw);
20200:     Configuration1', 'TMenuItem', iptrw);
20201:     MP3Player1', 'TMenuItem', iptrw);
20202:     DLLSpy1', 'TMenuItem', iptrw);
20203:     SynURIOpener1', 'TSynURIOpener', iptrw);
20204:     SynURISyn1', 'TSynURISyn', iptrw);
20205:     URILinksClicks1', 'TMenuItem', iptrw);
20206:     EditReplace1', 'TMenuItem', iptrw);
20207:     GotoLine1', 'TMenuItem', iptrw);
20208:     ActiveLineColor1', 'TMenuItem', iptrw);
20209:     ConfigFile1', 'TMenuItem', iptrw);
20210:     Sort1Intflist', 'TMenuItem', iptrw);
20211:     Redo1', 'TMenuItem', iptrw);
20212:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
20213:     Tutorial25Configuration1', 'TMenuItem', iptrw);
20214:     IndentSelection1', 'TMenuItem', iptrw);
20215:     UnindentSection1', 'TMenuItem', iptrw);
20216:     SkyStyle1', 'TMenuItem', iptrw);
20217:     N19', 'TMenuItem', iptrw);
20218:     CountWords1', 'TMenuItem', iptrw);
20219:     imbookmarkimages', 'TImageList', iptrw);
20220:     Bookmark11', 'TMenuItem', iptrw);
20221:     N20', 'TMenuItem', iptrw);
20222:     Bookmark21', 'TMenuItem', iptrw);
20223:     Bookmark31', 'TMenuItem', iptrw);
20224:     Bookmark41', 'TMenuItem', iptrw);
20225:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
20226:
20227:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
20228:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
20229:     Procedure PSScriptCompile( Sender : TPSScript)
20230:     Procedure Compile1Click( Sender : TObject)
20231:     Procedure PSScriptExecute( Sender : TPSScript)
20232:     Procedure open1Click( Sender : TObject)
20233:     Procedure Save2Click( Sender : TObject)
20234:     Procedure Savebefore1Click( Sender : TObject)
20235:     Procedure Largefont1Click( Sender : TObject)
20236:     Procedure FormActivate( Sender : TObject)
20237:     Procedure SBytecode1Click( Sender : TObject)
20238:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
20239:     Procedure Saveas3Click( Sender : TObject)
20240:     Procedure Clear1Click( Sender : TObject)
20241:     Procedure Slinenumbers1Click( Sender : TObject)
20242:     Procedure About1Click( Sender : TObject)
20243:     Procedure Search1Click( Sender : TObject)
20244:     Procedure FormCreate( Sender : TObject)
20245:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
20246:                                                     var Action : TSynReplaceAction)
20247:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
20248:     Procedure WordWrap1Click( Sender : TObject)
20249:     Procedure SearchNext1Click( Sender : TObject)
20250:     Procedure Replace1Click( Sender : TObject)
20251:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
20252:     Procedure ShowInclude1Click( Sender : TObject)
20253:     Procedure Printout1Click( Sender : TObject)
20254:     Procedure mnuPrintFont1Click( Sender : TObject)
20255:     Procedure Include1Click( Sender : TObject)
20256:     Procedure FormDestroy( Sender : TObject)
20257:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
20258:     Procedure UpdateView1Click( Sender : TObject)
20259:     Procedure CodeCompletionList1Click( Sender : TObject)
20260:     Procedure SaveOutput1Click( Sender : TObject)
```

```
20261:     Procedure ExportClipboard1Click( Sender : TObject)
20262:     Procedure Close1Click( Sender : TObject)
20263:     Procedure Manual1Click( Sender : TObject)
20264:     Procedure LoadLastFile1Click( Sender : TObject)
20265:     Procedure Memo1Change( Sender : TObject)
20266:     Procedure Decompile1Click( Sender : TObject)
20267:     Procedure StepInto1Click( Sender : TObject)
20268:     Procedure StepOut1Click( Sender : TObject)
20269:     Procedure Reset1Click( Sender : TObject)
20270:     Procedure cedebugAfterExecute( Sender : TPSScript)
20271:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
20272:     Procedure cedebugCompile( Sender : TPSScript)
20273:     Procedure cedebugExecute( Sender : TPSScript)
20274:     Procedure cedebugIdle( Sender : TObject)
20275:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
20276:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
20277:     Procedure BreakPointMenuClick( Sender : TObject)
20278:     Procedure DebugRun1Click( Sender : TObject)
20279:     Procedure tutorial4Click( Sender : TObject)
20280:     Procedure ImportfromClipboard1Click( Sender : TObject)
20281:     Procedure ImportfromClipboard2Click( Sender : TObject)
20282:     Procedure tutorial1Click( Sender : TObject)
20283:     Procedure ShowSpecChars1Click( Sender : TObject)
20284:     Procedure StatusBar1DblClick( Sender : TObject)
20285:     Procedure PSScriptLine( Sender : TObject)
20286:     Procedure OpenDirectory1Click( Sender : TObject)
20287:     Procedure procMessClick( Sender : TObject)
20288:     Procedure tbtnUseCaseClick( Sender : TObject)
20289:     Procedure EditFont1Click( Sender : TObject)
20290:     Procedure tutorial21Click( Sender : TObject)
20291:     Procedure tutorial31Click( Sender : TObject)
20292:     Procedure HTMLSyntax1Click( Sender : TObject)
20293:     Procedure ShowInterfaces1Click( Sender : TObject)
20294:     Procedure Tutorial5Click( Sender : TObject)
20295:     Procedure ShowLastException1Click( Sender : TObject)
20296:     Procedure PlayMP31Click( Sender : TObject)
20297:     Procedure AllFunctionsList1Click( Sender : TObject)
20298:     Procedure texSyntax1Click( Sender : TObject)
20299:     Procedure GetEMails1Click( Sender : TObject)
20300:     procedure DelphiSite1Click(Sender: TObject);
20301:     procedure TerminalStyle1Click(Sender: TObject);
20302:     procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
20303:     procedure ShellStyle1Click(Sender: TObject);
20304:     procedure Console1Click(Sender: TObject);     //3.2
20305:     procedure BigScreen1Click(Sender: TObject);
20306:     procedure Tutorial91Click(Sender: TObject);
20307:     procedure SaveScreenshotClick(Sender: TObject);
20308:     procedure Tutorial101Click(Sender: TObject);
20309:     procedure SQLSyntax1Click(Sender: TObject);
20310:     procedure XMLSyntax1Click(Sender: TObject);
20311:     procedure ComponentCount1Click(Sender: TObject);
20312:     procedure NewInstance1Click(Sender: TObject);
20313:     procedure CSyntax1Click(Sender: TObject);
20314:     procedure Tutorial6Click(Sender: TObject);
20315:     procedure New1Click(Sender: TObject);
20316:     procedure AllObjectsList1Click(Sender: TObject);
20317:     procedure LoadBytecode1Click(Sender: TObject);
20318:     procedure CipherFile1Click(Sender: TObject);  //V3.5
20319:     procedure NewInstance1Click(Sender: TObject);
20320:     procedure toolbtnTutorialClick(Sender: TObject);
20321:     procedure Memory1Click(Sender: TObject);
20322:     procedure JavaSyntax1Click(Sender: TObject);
20323:     procedure SyntaxCheck1Click(Sender: TObject);
20324:     procedure ScriptExplorer1Click(Sender: TObject);
20325:     procedure FormOutput1Click(Sender: TObject);  //V3.6
20326:     procedure GotoEnd1Click(Sender: TObject);
20327:     procedure AllResourceList1Click(Sender: TObject);
20328:     procedure tbtn6resClick(Sender: TObject);   //V3.7
20329:     procedure Info1Click(Sender: TObject);
20330:     procedure Tutorial10Statistics1Click(Sender: TObject);
20331:     procedure Tutorial11Forms1Click(Sender: TObject);
20332:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
20333:     procedure ResourceExplore1Click(Sender: TObject);
20334:     procedure Info1Click(Sender: TObject);
20335:     procedure CryptoBox1Click(Sender: TObject);
20336:     procedure ModulesCount1Click(Sender: TObject);
20337:     procedure N4GewinntGame1Click(Sender: TObject);
20338:     procedure PHPSyntax1Click(Sender: TObject);
20339:     procedure SerialRS2321Click(Sender: TObject);
20340:     procedure CSyntax2Click(Sender: TObject);
20341:     procedure Calculator1Click(Sender: TObject);
20342:     procedure Tutorial13Ciphering1Click(Sender: TObject);
20343:     procedure Tutorial14Async1Click(Sender: TObject);
20344:     procedure PHPSyntax1Click(Sender: TObject);
20345:     procedure BtnZoomPlusClick(Sender: TObject);
20346:     procedure BtnZoomMinusClick(Sender: TObject);
20347:     procedure btnClassReportClick(Sender: TObject);
20348:     procedure ThreadDemo1Click(Sender: TObject);
20349:     procedure HEXView1Click(Sender: TObject);
```

```
20350:     procedure ExporttoHTML1Click(Sender: TObject);
20351:     procedure Minesweeper1Click(Sender: TObject);
20352:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
20353:     procedure sbvclhelpClick(Sender: TObject);
20354:     procedure DependencyWalker1Click(Sender: TObject);
20355:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
20356:     procedure WebScanner1Click(Sender: TObject);
20357:     procedure mnToolbar1Click(Sender: TObject);
20358:     procedure mnStatusbar2Click(Sender: TObject);
20359:     procedure mnConsole2Click(Sender: TObject);
20360:     procedure mnCoolbar2Click(Sender: TObject);
20361:     procedure mnSplitter2Click(Sender: TObject);
20362:     procedure WebServer1Click(Sender: TObject);
20363:     procedure PerlSyntax1Click(Sender: TObject);
20364:     procedure PythonSyntax1Click(Sender: TObject);
20365:     procedure DMathLibrary1Click(Sender: TObject);
20366:     procedure IntfNavigator1Click(Sender: TObject);
20367:     procedure FullTextFinder1Click(Sender: TObject);
20368:     function AppName: string;
20369:     function ScriptName: string;
20370:     function LastName: string;
20371:     procedure FractalDemo1Click(Sender: TObject);
20372:     procedure SimuLogBox1Click(Sender: TObject);
20373:     procedure OpenExamples1Click(Sender: TObject);
20374:     procedure Halt1Click(Sender: TObject);
20375:     procedure Stop;
20376:     procedure CodeSearch1Click(Sender: TObject);
20377:     procedure RubySyntax1Click(Sender: TObject);
20378:     procedure Undo1Click(Sender: TObject);
20379:     procedure LinuxShellScript1Click(Sender: TObject);
20380:     procedure WebScannerDirect(urls: string);
20381:     procedure WebScanner(urls: string);
20382:     procedure LoadInterfaceList2;
20383:     procedure DLLSpy1Click(Sender: TObject);
20384:     procedure Memo1DblClick(Sender: TObject);
20385:     procedure URILinksClicks1Click(Sender: TObject);
20386:     procedure GotoLine1Click(Sender: TObject);
20387:     procedure ConfigFile1Click(Sender: TObject);
20388:     Procedure Sort1IntflistClick( Sender : TObject)
20389:     Procedure Redo1Click( Sender : TObject)
20390:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
20391:     Procedure IndentSelection1Click( Sender : TObject)
20392:     Procedure UnindentSection1Click( Sender : TObject)
20393:     Procedure SkyStyle1Click( Sender : TObject)
20394:     Procedure CountWords1Click( Sender : TObject)
20395:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
20396:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
20397:     Procedure Bookmark11Click( Sender : TObject)
20398:     Procedure Bookmark21Click( Sender : TObject)
20399:     Procedure Bookmark31Click( Sender : TObject)
20400:     Procedure Bookmark41Click( Sender : TObject)
20401:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
20402:     'STATMemoryReport', 'boolean', iptrw);
20403:     'IPPort', 'integer', iptrw);
20404:     'COMPort', 'integer', iptrw);
20405:     'lbintflist', 'TListBox', iptrw);
20406:     Function GetStatChange : boolean
20407:     Procedure SetStatChange( vstat : boolean)
20408:     Function GetActFileName : string
20409:     Procedure SetActFileName( vname : string)
20410:     Function GetLastFileName : string
20411:     Procedure SetLastFileName( vname : string)
20412:     Procedure WebScannerDirect( urls : string)
20413:     Procedure LoadInterfaceList2
20414:     Function GetStatExecuteShell : boolean
20415:     Procedure DoEditorExecuteCommand( EditorCommand : word)
20416:     function GetActiveLineColor: TColor
20417:     procedure SetActiveLineColor(acolor: TColor)
20418:     procedure ScriptListbox1Click(Sender: TObject);
20419:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
20420:     procedure EnlargeGutter1Click(Sender: TObject);
20421:     procedure Tetris1Click(Sender: TObject);
20422:     procedure ToDoList1Click(Sender: TObject);
20423:     procedure ProcessList1Click(Sender: TObject);
20424:     procedure MetricReport1Click(Sender: TObject);
20425:     procedure ProcessList1Click(Sender: TObject);
20426:     procedure TCPSockets1Click(Sender: TObject);
20427:     procedure ConfigUpdate1Click(Sender: TObject);
20428:     procedure ADOWorkbench1Click(Sender: TObject);
20429:     procedure SocketServer1Click(Sender: TObject);
20430:     procedure FormDemo1Click(Sender: TObject);
20431:     procedure Richedit1Click(Sender: TObject);
20432:     procedure SimpleBrowser1Click(Sender: TObject);
20433:     procedure DOSShell1Click(Sender: TObject);
20434:     procedure SynExport1Click(Sender: TObject);
20435:     procedure ExporttoRTF1Click(Sender: TObject);
20436:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
20437:     procedure SOAPTester1Click(Sender: TObject);
20438:     procedure Sniffer1Click(Sender: TObject);
```

```
20439:     procedure AutoDetectSyntax1Click(Sender: TObject);
20440:     procedure FPlot1Click(Sender: TObject);
20441:     procedure PasStyle1Click(Sender: TObject);
20442:     procedure Tutorial183RGBLED1Click(Sender: TObject);
20443:     procedure Reversi1Click(Sender: TObject);
20444:     procedure ManualmaXbox1Click(Sender: TObject);
20445:     procedure BlaisePascalMagazine1Click(Sender: TObject);
20446:     procedure AddToDo1Click(Sender: TObject);
20447:     procedure CreateGUID1Click(Sender: TObject);
20448:     procedure Tutorial27XML1Click(Sender: TObject);
20449:     procedure CreateDLLStub1Click(Sender: TObject);
20450:     procedure Tutorial28DLL1Click(Sender: TObject);');
20451:     procedure ResetKeyPressed;');
20452:     procedure KeyPressedFalse;
20453:     procedure FileChanges1Click(Sender: TObject);');
20454:     procedure OpenGLTry1Click(Sender: TObject);');
20455:     procedure AllUnitList1Click(Sender: TObject);');
20456:     procedure Tutorial29UMLClick(Sender: TObject);
20457:     procedure CreateHeader1Click(Sender: TObject);
20458:     procedure Oscilloscope1Click(Sender: TObject);');
20459:     procedure Tutorial30WOT1Click(Sender: TObject);');
20460:     procedure GetWebScript1Click(Sender: TObject);');
20461:     procedure Checkers1Click(Sender: TObject);');
20462:     procedure TaskMgr1Click(Sender: TObject);');
20463:     procedure WebCam1Click(Sender: TObject);');
20464:
20465:
20466: //----------------------------------------------------------------------------
20467: //*************mX4 Editor SynEdit Tools API *********************************
20468: //----------------------------------------------------------------------------
20469: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
20470: begin
20471:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
20472:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
20473:     Constructor Create( AOwner : TComponent)
20474:     SelStart', 'Integer', iptrw);
20475:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
20476:     Procedure UpdateCaret
20477:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
20478:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
20479:     Procedure BeginUndoBlock
20480:     Procedure BeginUpdate
20481:     Function CaretInView : Boolean
20482:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
20483:     Procedure Clear
20484:     Procedure ClearAll
20485:     Procedure ClearBookMark( BookMark : Integer)
20486:     Procedure ClearSelection
20487:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
20488:     Procedure ClearUndo
20489:     Procedure CopyToClipboard
20490:     Procedure CutToClipboard
20491:     Procedure DoCopyToClipboard( const SText : string)
20492:     Procedure EndUndoBlock
20493:     Procedure EndUpdate
20494:     Procedure EnsureCursorPosVisible
20495:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
20496:     Procedure FindMatchingBracket
20497:     Function GetMatchingBracket : TBufferCoord
20498:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
20499:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
20500:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
20501:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
20502:               : TSynHighlighterAttributes) : boolean
20503:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
20504:               var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
20505:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
20506:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
20507:     Procedure GotoBookMark( BookMark : Integer)
20508:     Procedure GotoLineAndCenter( ALine : Integer)
20509:     Function IdentChars : TSynIdentChars
20510:     Procedure InvalidateGutter
20511:     Procedure InvalidateGutterLine( aLine : integer)
20512:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
20513:     Procedure InvalidateLine( Line : integer)
20514:     Procedure InvalidateLines( FirstLine, LastLine : integer)
20515:     Procedure InvalidateSelection
20516:     Function IsBookmark( BookMark : integer) : boolean
20517:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
20518:     Procedure LockUndo
20519:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
20520:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
20521:     Function LineToRow( aLine : integer) : integer
20522:     Function RowToLine( aRow : integer) : integer
20523:     Function NextWordPos : TBufferCoord
20524:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
20525:     Procedure PasteFromClipboard
20526:     Function WordStart : TBufferCoord
20527:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
```

```
20528:     Function WordEnd : TBufferCoord
20529:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
20530:     Function PrevWordPos : TBufferCoord
20531:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
20532:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
20533:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
20534:     Procedure Redo
20535:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
20536:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
20537:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
20538:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
20539:     Procedure SelectAll
20540:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
20541:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
20542:     Procedure SetDefaultKeystrokes
20543:     Procedure SetSelWord
20544:     Procedure SetWordBlock( Value : TBufferCoord)
20545:     Procedure Undo
20546:     Procedure UnlockUndo
20547:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
20548:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
20549:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
20550:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
20551:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
20552:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
20553:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
20554:     Procedure AddFocusControl( aControl : TWinControl)
20555:     Procedure RemoveFocusControl( aControl : TWinControl)
20556:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
20557:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
20558:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
20559:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
20560:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
20561:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
20562:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
20563:     Procedure RemoveLinesPointer
20564:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
20565:     Procedure UnHookTextBuffer
20566:     BlockBegin', 'TBufferCoord', iptrw);
20567:     BlockEnd', 'TBufferCoord', iptrw);
20568:     CanPaste', 'Boolean', iptr);
20569:     CanRedo', 'boolean', iptr);
20570:     CanUndo', 'boolean', iptr);
20571:     CaretX', 'Integer', iptrw);
20572:     CaretY', 'Integer', iptrw);
20573:     CaretXY', 'TBufferCoord', iptrw);
20574:     ActiveLineColor', 'TColor', iptrw);
20575:     DisplayX', 'Integer', iptr);
20576:     DisplayY', 'Integer', iptr);
20577:     DisplayXY', 'TDisplayCoord', iptr);
20578:     DisplayLineCount', 'integer', iptr);
20579:     CharsInWindow', 'Integer', iptr);
20580:     CharWidth', 'integer', iptr);
20581:     Font', 'TFont', iptrw);
20582:     GutterWidth', 'Integer', iptr);
20583:     Highlighter', 'TSynCustomHighlighter', iptrw);
20584:     LeftChar', 'Integer', iptrw);
20585:     LineHeight', 'integer', iptr);
20586:     LinesInWindow', 'Integer', iptr);
20587:     LineText', 'string', iptrw);
20588:     Lines', 'TStrings', iptrw);
20589:     Marks', 'TSynEditMarkList', iptr);
20590:     MaxScrollWidth', 'integer', iptrw);
20591:     Modified', 'Boolean', iptrw);
20592:     PaintLock', 'Integer', iptr);
20593:     ReadOnly', 'Boolean', iptrw);
20594:     SearchEngine', 'TSynEditSearchCustom', iptrw);
20595:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
20596:     SelTabBlock', 'Boolean', iptr);
20597:     SelTabLine', 'Boolean', iptr);
20598:     SelText', 'string', iptrw);
20599:     StateFlags', 'TSynStateFlags', iptr);
20600:     Text', 'string', iptrw);
20601:     TopLine', 'Integer', iptrw);
20602:     WordAtCursor', 'string', iptr);
20603:     WordAtMouse', 'string', iptr);
20604:     UndoList', 'TSynEditUndoList', iptr);
20605:     RedoList', 'TSynEditUndoList', iptr);
20606:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
20607:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
20608:     BorderStyle', 'TSynBorderStyle', iptrw);
20609:     ExtraLineSpacing', 'integer', iptrw);
20610:     Gutter', 'TSynGutter', iptrw);
20611:     HideSelection', 'boolean', iptrw);
20612:     InsertCaret', 'TSynEditCaretType', iptrw);
20613:     InsertMode', 'boolean', iptrw);
20614:     IsScrolling', 'Boolean', iptr);
20615:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
20616:     MaxUndo', 'Integer', iptrw);
```

```
20617:     Options', 'TSynEditorOptions', iptrw);
20618:     OverwriteCaret', 'TSynEditCaretType', iptrw);
20619:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
20620:     ScrollHintColor', 'TColor', iptrw);
20621:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
20622:     ScrollBars', 'TScrollStyle', iptrw);
20623:     SelectedColor', 'TSynSelectedColor', iptrw);
20624:     SelectionMode', 'TSynSelectionMode', iptrw);
20625:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
20626:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
20627:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
20628:     WordWrapGlyph', 'TSynGlyph', iptrw);
20629:     OnChange', 'TNotifyEvent', iptrw);
20630:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
20631:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
20632:     OnContextHelp', 'TContextHelpEvent', iptrw);
20633:     OnDropFiles', 'TDropFilesEvent', iptrw);
20634:     OnGutterClick', 'TGutterClickEvent', iptrw);
20635:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
20636:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
20637:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
20638:     OnPaint', 'TPaintEvent', iptrw);
20639:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
20640:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
20641:     OnReplaceText', 'TReplaceTextEvent', iptrw);
20642:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
20643:     OnStatusChange', 'TStatusChangeEvent', iptrw);
20644:     OnPaintTransient', 'TPaintTransient', iptrw);
20645:     OnScroll', 'TScrollEvent', iptrw);
20646:   end;
20647:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
20648:  Function GetPlaceableHighlighters : TSynHighlighterList
20649:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
20650:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
20651:  Procedure GetEditorCommandValues( Proc : TGetStrProc)
20652:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)
20653:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
20654:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
20655:  Function ConvertCodeStringToExtended( AString : String) : String
20656:  Function ConvertExtendedToCodeString( AString : String) : String
20657:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
20658:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
20659:  Function IndexToEditorCommand( const AIndex : Integer) : Integer
20660:
20661:   TSynEditorOption = (
20662:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
20663:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
20664:                                // preceding line
20665:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
20666:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
20667:                                //direction any more
20668:     eoDragDropEditing,         //Allows to select a textblock and drag it in document to another location
20669:     eoDropFiles,               //Allows the editor accept OLE file drops
20670:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
20671:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
20672:     eoGroupUndo,               //When undoing/redoing actions,handle all cont.changes same kind in onecall
20673:                                //instead undoing/redoing each command separately
20674:     eoHalfPageScroll,          //By scrolling with page-up/page-down commands,only scroll half page attime
20675:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
20676:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
20677:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
20678:     eoNoCaret,                 //Makes it so the caret is never visible
20679:     eoNoSelection,             //Disables selecting text
20680:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
20681:     eoScrollByOneLess,         //Forces scrolling to be one less
20682:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
20683:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
20684:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
20685:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
20686:     eoShowSpecialChars,        //Shows the special Characters
20687:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
20688:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
20689:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
20690:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
20691:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
20692:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
20693:
20694:     ********************************Important Editor Short Cuts*****************************);
20695:     Double click to select a word and count words with highlightning.
20696:     Triple click to select a line.
20697:     CTRL+SHIFT+click to extend a selection.
20698:     Drag with the ALT key down to select columns of text !!!
20699:     Drag and drop is supported.
20700:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
20701:     Type CTRL+A to select all.
20702:     Type CTRL+N to set a new line.
20703:     Type CTRL+T to delete a line or token. //Tokenizer
20704:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
20705:     Type CTRL+Shift+T to add ToDo in line and list.
```

```
20706:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
20707:     Type CTRL[0..9] to jump or get to bookmarks.
20708:     Type Home to position cursor at beginning of current line and End to position it at end of line.
20709:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
20710:     Page Up and Page Down work as expected.
20711:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
20712:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
20713:
20714: {$ Short Key Positions Ctrl<A-Z>: }
20715: def
20716:    <A> Select All
20717:    <B> Count Words
20718:    <C> Copy
20719:    <D> Internet Start
20720:    <E> Script List
20721:    <F> Find
20722:    <G> Goto
20723:    <H> Mark Line
20724:    <I> Interface List
20725:    <J> Code Completion
20726:    <K> Console
20727:    <L> Interface List Box
20728:    <M> Font Larger -
20729:    <N> New Line
20730:    <O> Open File
20731:    <P> Font Smaller +
20732:    <Q> Quit
20733:    <R> Replace
20734:    <S> Save!
20735:    <T> Delete Line
20736:    <U> Use Case Editor
20737:    <V> Paste
20738:    <W> URI Links
20739:    <X> Reserved for coding use internal
20740:    <Y> Delete Line
20741:    <Z> Undo
20742:
20743: ref
20744:    F1 Help
20745:    F2 Syntax Check
20746:    F3 Search Next
20747:    F4 New Instance
20748:    F5 Line Mark /Breakpoint
20749:    F6 Goto End
20750:    F7 Debug Step Into
20751:    F8 Debug Step Out
20752:    F9 Compile
20753:    F10 Menu
20754:    F11 Word Count Highlight
20755:    F12 Reserved for coding use internal
20756:
20757:    AddRegisteredVariable( it ,integer');  //for closure!!
20758:    AddRegisteredVariable( sr ,string');  //for closure
20759:    AddRegisteredVariable( bt ,boolean');  //for closure
20760:    AddRegisteredVariable( ft ,double');  //for closure
20761:    AddRegisteredVariable( srlist ,TStringlist');  //for closures
20762:
20763:  def ReservedWords: array[0..82] of string =
20764:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
20765:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
20766:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
20767:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
20768:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
20769:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
20770:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
20771:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
20772:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
20773:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
20774:      'public', 'published',def,ref,using,typedef ,memo1','memo2','doc','maxform1','it';
20775:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
20776: //-------------------------------------------------------------------------
20777: //*************End of mX4 Public  Tools API *********************************
20778: //-------------------------------------------------------------------------
20779:
20780: Amount of Functions: 13271
20781: Amount of Procedures: 8274
20782: Amount of Constructors: 1338
20783: Totals of Calls: 22883
20784: SHA1: Win 3.9.9.98 D0EC95326FE1ABD9D441F137336D00CF4BC77CAB
20785:
20786:
20787: ************************************************************
20788:  Doc Short Manual with 50 Tips!
20789: ************************************************************
20790: - Install: just save your maxboxdef.ini before and then extract the zip file!
20791: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
20792: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
20793: - Menu: With <Crtl><F3> you can search for code on examples
20794: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
```

```
20795: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
20796: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
20797:
20798: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
20799: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
20800: - Context Menu: You can printout your scripts as a pdf-file or html-export
20801: - Context: You do have a context menu with the right mouse click
20802:
20803: - Menu: With the UseCase Editor you can convert graphic formats too.
20804: - Menu: On menu Options you find Addons as compiled scripts
20805: - IDE: You don't need a mouse to handle maXbox, use shortcuts
20806: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
20807: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
20808: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
20809:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
20810:
20811: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
20812: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
20813: - Code: If you code a loop till key-pressed use function: isKeyPressed;
20814: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
20815: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
20816: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
20817:          to delete and Click and mark to drag a bookmark
20818: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
20819: - IDE: A file info with system and script information you find in menu Program/Information
20820: - IDE: After change the config file in help you can update changes in menu Help/Config Update
20821: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
20822: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
20823: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
20824: - Editor: Set Bookmarks to check your work in app or code
20825: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
20826: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
20827: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
20828:
20829: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
20830: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
20831: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
20832: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
20833: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
20834: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
20835: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
20836: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
20837: - IDE menu /Help/Tools/ open the Task Manager
20838:
20839: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
20840: - Add on when no browser is available start /Options/Add ons/Easy Browser
20841: - Add on SOAP Tester with SOP POST File
20842: - Add on IP Protocol Sniffer with List View
20843: - Add on OpenGL mX Robot Demo for android
20844: - Add on Checkers Game
20845:   Add on Oscilloscope
20846:
20847: - Menu: Help/Tools as a Tool Section with DOS Opener
20848: - Menu Editor: export the code as RTF File
20849: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
20850: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
20851: - Context: Auto Detect of Syntax depending on file extension
20852: - Code: some Windows API function start with w in the name like wGetAtomName();
20853: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
20854: - IDE File Check with menu ..View/File Changes/...
20855: - Context: Create a Header with Create Header in Navigator List at right window
20856: - Code: use  SysErrorMessage to get a real Error Description, Ex.
20857:         RemoveDir('c:\NoSuchFolder') ;
20858:         writeln('System Error Message: '+ SysErrorMessage(GetLastError));
20859: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
20860:
20861: - using DLL example in maXbox: //function: {*****************************************}
20862:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
20863:                                  cb: DWORD): BOOL; //stdcall;;
20864:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
20865:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
20866:      External  'OpenProcess@kernel32.dll stdcall';
20867:
20868:
20869: GCC Compile Ex Script
20870: procedure TFormMain_btnCompileClick(Sender: TObject);
20871: begin
20872:   AProcess:= TProcess.Create(Nil);
20873:   try  AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
20874:       + ' -o "' + OpenDialog2.FileName + '"';
20875:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
20876:     AProcess.Execute;
20877:     Memo2.Lines.BeginUpdate;
20878:     Memo2.Lines.Clear;
20879:     Memo2.Lines.LoadFromStream(AProcess.Output);
20880:     Memo2.Lines.EndUpdate;
20881:   finally
20882:     AProcess.Free;
20883:   end;
```

```
20884: end;
20885:
20886: Stopwatch pattern
20887: Time1:= Time;
20888:   writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
20889:   if initAndStartBoard then
20890:      writeln('Filesize: '+inttoStr(filesize(FILESAVE)));
20891:   writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
20892:   PrintF('%d %s',[Trunc((Time-Time1)*24),
20893:                   FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
20894:
20895:    POST git-receive-pack (chunked)
20896: Pushing to https://github.com/maxkleiner/maXbox3.git
20897: To https://github.com/maxkleiner/maXbox3.git
20898:    f127d21..c6a98da  masterbox2 -> masterbox2
20899: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
20900:
20901: History Shell Hell
20902: PCT Precompile Technology , mX4 ScriptStudio
20903: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
20904: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
20905: emax layers: system-package-component-unit-class-function-block
20906: new keywords def ref using maXCalcF
20907: UML: use case act class state seq pac comp dep - lib lab
20908: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
20909: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
20910: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
20911: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
20912: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
20913: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
20914: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
20915: SendBuffer, Color+Caption hack, ComboSet,SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
20916: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
20917: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
20918: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
20919: Inno Install and Setup Routines
20920: Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
20921: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
20922: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
20923: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
20924:
20925:
20926: Ref:
20927: https://unibe-ch.academia.edu/MaxKleiner
20928: http://www.slideshare.net/maxkleiner1
20929: http://www.scribd.com/max_kleiner
20930: http://www.delphiforfun.org/Programs/Utilities/index.htm
20931: http://www.slideshare.net/maxkleiner1
20932: http://s3.amazonaws.com/PreviewLinks/22959.html
20933: http://www.softwareschule.ch/arduino_training.pdf
20934: http://www.jrsoftware.org/isinfo.php
20935: http://www.be-precision.com/products/precision-builder/express/
20936: http://www.blaisepascal.eu/
20937: http://www.delphibasics.co.uk/
20938: http://www.youtube.com/watch?v=av89HAbqAsI
20939: http://www.angelfire.com/hi5/delphizeus/modal.html
20940: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
20941: http://delphi.org/2014/01/every-android-api-for-delphi/
20942: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
20943:
20944:
20945:
20946:
20947:
20948: ************************************************************
20949:  unit List asm internal end
20950: ************************************************************
20951: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
20952: 02 unit SIRegister_IdStrings                   //Indy Sockets
20953: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
20954: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
20955: 05 unit IFSI_WinForm1puzzle;                   //maXbox
20956: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
20957: 07 unit RegisterDateTimeLibrary_R(exec);      //Delphi
20958: 08 unit RIRegister_MathMax_Routines(exec);    //Jedi & Delphi
20959: 09 unit RIRegister_IdGlobal_Routines(exec);   //Indy Sockets
20960: 10 unit RIRegister_SysUtils_Routines(Exec);   //Delphi
20961: 11 unit uPSI_IdTCPConnection;                 //Indy some functions
20962: 12 unit uPSCompiler.pas;                      //PS kernel functions
20963: 13 unit uPSI_DBCommon;                        //DB Common_Routines and Types
20964: 14 unit uPSI_Printers.pas                     //Delphi VCL
20965: 15 unit uPSI_MPlayer.pas                     //Delphi VCL
20966: 16 unit uPSC_comobj;                         //COM Functions
20967: 17 unit uPSI_Clipbrd;                        //Delphi VCL
20968: 18 unit Filectrl in IFSI_SysUtils_max;      //VCL Runtime
20969: 19 unit uPSI_SqlExpr;                       //DBX3
20970: 20 unit uPSI_ADODB;                         //ADODB
20971: 21 unit uPSI_StrHlpr;                       //String Helper Routines
20972: 22 unit uPSI_DateUtils;                     //Expansion to DateTimeLib
```

```
20973: 23 unit uPSI_FileUtils;                    //Expansion to Sys/File Utils
20974: 24 unit JUtils / gsUtils;                  //Jedi / Metabase
20975: 25 unit JvFunctions_max;                   //Jedi Functions
20976: 26 unit HTTPParser;                        //Delphi VCL
20977: 27 unit HTTPUtil;                         //Delphi VCL
20978: 28 unit uPSI_XMLUtil;                     //Delphi VCL
20979: 29 unit uPSI_SOAPHTTPClient;              //Delphi VCL SOAP WebService V3.5
20980: 30 unit uPSI_Contnrs;                     //Delphi RTL Container of Classes
20981: 31 unit uPSI_MaskUtils;                   //RTL Edit and Mask functions
20982: 32 unit uPSI_MyBigInt;                    //big integer class with Math
20983: 33 unit uPSI_ConvUtils;                   //Delphi VCL Conversions engine
20984: 34 unit Types_Variants;                   //Delphi\Win32\rtl\sys
20985: 35 unit uPSI_IdHashSHA1;                  //Indy Crypto Lib
20986: 36 unit uPSI_IdHashMessageDigest          //Indy Crypto;
20987: 37 unit uPSI_IdASN1Util;                  //Indy ASN1Utility Routines;
20988: 38 unit uPSI_IdLogFile;                   //Indy Logger from LogBase
20989: 39 unit uPSI_IdIcmpClient;                //Indy Ping ICMP
20990: 40 unit uPSI_IdHashMessageDigest_max      //Indy Crypto &OpenSSL;
20991: 41 unit uPSI_FileCtrl;                    //Delphi RTL
20992: 42 unit uPSI_Outline;                     //Delphi VCL
20993: 43 unit uPSI_ScktComp;                    //Delphi RTL
20994: 44 unit uPSI_Calendar;                    //Delphi VCL
20995: 45 unit uPSI_VListView                    //VListView;
20996: 46 unit uPSI_DBGrids;                     //Delphi VCL
20997: 47 unit uPSI_DBCtrls;                     //Delphi VCL
20998: 48 unit ide_debugoutput;                  //maXbox
20999: 49 unit uPSI_ComCtrls;                   //Delphi VCL
21000: 50 unit uPSC_stdctrls+;                  //Delphi VCL
21001: 51 unit uPSI_Dialogs;                    //Delphi VCL
21002: 52 unit uPSI_StdConvs;                   //Delphi RTL
21003: 53 unit uPSI_DBClient;                   //Delphi RTL
21004: 54 unit uPSI_DBPlatform;                 //Delphi RTL
21005: 55 unit uPSI_Provider;                   //Delphi RTL
21006: 56 unit uPSI_FMTBcd;                     //Delphi RTL
21007: 57 unit uPSI_DBCGrids;                   //Delphi VCL
21008: 58 unit uPSI_CDSUtil;                    //MIDAS
21009: 59 unit uPSI_VarHlpr;                    //Delphi RTL
21010: 60 unit uPSI_ExtDlgs;                    //Delphi VCL
21011: 61 unit sdpStopwatch;                    //maXbox
21012: 62 unit uPSI_JclStatistics;              //JCL
21013: 63 unit uPSI_JclLogic;                   //JCL
21014: 64 unit uPSI_JclMiscel;                  //JCL
21015: 65 unit uPSI_JclMath_max;                //JCL RTL
21016: 66 unit uPSI_uTPLb_StreamUtils;          //LockBox 3
21017: 67 unit uPSI_MathUtils;                  //BCB
21018: 68 unit uPSI_JclMultimedia;              //JCL
21019: 69 unit uPSI_WideStrUtils;               //Delphi API/RTL
21020: 70 unit uPSI_GraphUtil;                  //Delphi RTL
21021: 71 unit uPSI_TypeTrans;                  //Delphi RTL
21022: 72 unit uPSI_HTTPApp;                    //Delphi VCL
21023: 73 unit uPSI_DBWeb;                     //Delphi VCL
21024: 74 unit uPSI_DBBdeWeb;                  //Delphi VCL
21025: 75 unit uPSI_DBXpressWeb;               //Delphi VCL
21026: 76 unit uPSI_ShadowWnd;                 //Delphi VCL
21027: 77 unit uPSI_ToolWin;                   //Delphi VCL
21028: 78 unit uPSI_Tabs;                      //Delphi VCL
21029: 79 unit uPSI_JclGraphUtils;             //JCL
21030: 80 unit uPSI_JclCounter;                //JCL
21031: 81 unit uPSI_JclSysInfo;                //JCL
21032: 82 unit uPSI_JclSecurity;               //JCL
21033: 83 unit uPSI_JclFileUtils;              //JCL
21034: 84 unit uPSI_IdUserAccounts;            //Indy
21035: 85 unit uPSI_IdAuthentication;          //Indy
21036: 86 unit uPSI_uTPLb_AES;                 //LockBox 3
21037: 87 unit uPSI_IdHashSHA1;                //LockBox 3
21038: 88 unit uTPLb_BlockCipher;              //LockBox 3
21039: 89 unit uPSI_ValEdit.pas;               //Delphi VCL
21040: 90 unit uPSI_JvVCLUtils;                //JCL
21041: 91 unit uPSI_JvDBUtil;                  //JCL
21042: 92 unit uPSI_JvDBUtils;                 //JCL
21043: 93 unit uPSI_JvAppUtils;                //JCL
21044: 94 unit uPSI_JvCtrlUtils;               //JCL
21045: 95 unit uPSI_JvFormToHtml;              //JCL
21046: 96 unit uPSI_JvParsing;                 //JCL
21047: 97 unit uPSI_SerDlgs;                   //Toolbox
21048: 98 unit uPSI_Serial;                    //Toolbox
21049: 99 unit uPSI_JvComponent;               //JCL
21050: 100 unit uPSI_JvCalc;                   //JCL
21051: 101 unit uPSI_JvBdeUtils;               //JCL
21052: 102 unit uPSI_JvDateUtil;               //JCL
21053: 103 unit uPSI_JvGenetic;                //JCL
21054: 104 unit uPSI_JclBase;                  //JCL
21055: 105 unit uPSI_JvUtils;                  //JCL
21056: 106 unit uPSI_JvStrUtil;                //JCL
21057: 107 unit uPSI_JvStrUtils;               //JCL
21058: 108 unit uPSI_JvFileUtil;               //JCL
21059: 109 unit uPSI_JvMemoryInfos;            //JCL
21060: 110 unit uPSI_JvComputerInfo;           //JCL
21061: 111 unit uPSI_JvgCommClasses;           //JCL
```

```
21062: 112 unit uPSI_JvgLogics;                    //JCL
21063: 113 unit uPSI_JvLED;                        //JCL
21064: 114 unit uPSI_JvTurtle;                     //JCL
21065: 115 unit uPSI_SortThds; unit uPSI_ThSort;   //maXbox
21066: 116 unit uPSI_JvgUtils;                     //JCL
21067: 117 unit uPSI_JvExprParser;                 //JCL
21068: 118 unit uPSI_HexDump;                      //Borland
21069: 119 unit uPSI_DBLogDlg;                     //VCL
21070: 120 unit uPSI_SqlTimSt;                     //RTL
21071: 121 unit uPSI_JvHtmlParser;                 //JCL
21072: 122 unit uPSI_JvgXMLSerializer;             //JCL
21073: 123 unit uPSI_JvJCLUtils;                   //JCL
21074: 124 unit uPSI_JvStrings;                    //JCL
21075: 125 unit uPSI_uTPLb_IntegerUtils;           //TurboPower
21076: 126 unit uPSI_uTPLb_HugeCardinal;           //TurboPower
21077: 127 unit uPSI_uTPLb_HugeCardinalUtils;      //TurboPower
21078: 128 unit uPSI_SynRegExpr;                   //SynEdit
21079: 129 unit uPSI_StUtils;                      //SysTools4
21080: 130 unit uPSI_StToHTML;                     //SysTools4
21081: 131 unit uPSI_StStrms;                      //SysTools4
21082: 132 unit uPSI_StFIN;                        //SysTools4
21083: 133 unit uPSI_StAstroP;                     //SysTools4
21084: 134 unit uPSI_StStat;                       //SysTools4
21085: 135 unit uPSI_StNetCon;                     //SysTools4
21086: 136 unit uPSI_StDecMth;                     //SysTools4
21087: 137 unit uPSI_StOStr;                       //SysTools4
21088: 138 unit uPSI_StPtrns;                      //SysTools4
21089: 139 unit uPSI_StNetMsg;                     //SysTools4
21090: 140 unit uPSI_StMath;                       //SysTools4
21091: 141 unit uPSI_StExpEng;                     //SysTools4
21092: 142 unit uPSI_StCRC;                        //SysTools4
21093: 143 unit uPSI_StExport,                     //SysTools4
21094: 144 unit uPSI_StExpLog,                     //SysTools4
21095: 145 unit uPSI_ActnList;                     //Delphi VCL
21096: 146 unit uPSI_jpeg;                         //Borland
21097: 147 unit uPSI_StRandom;                     //SysTools4
21098: 148 unit uPSI_StDict;                       //SysTools4
21099: 149 unit uPSI_StBCD;                        //SysTools4
21100: 150 unit uPSI_StTxtDat;                     //SysTools4
21101: 151 unit uPSI_StRegEx;                      //SysTools4
21102: 152 unit uPSI_IMouse;                       //VCL
21103: 153 unit uPSI_SyncObjs;                     //VCL
21104: 154 unit uPSI_AsyncCalls;                   //Hausladen
21105: 155 unit uPSI_ParallelJobs;                 //Saraiva
21106: 156 unit uPSI_Variants;                     //VCL
21107: 157 unit uPSI_VarCmplx;                     //VCL Wolfram
21108: 158 unit uPSI_DTDSchema;                    //VCL
21109: 159 unit uPSI_ShLwApi;                      //Brakel
21110: 160 unit uPSI_IBUtils;                      //VCL
21111: 161 unit uPSI_CheckLst;                     //VCL
21112: 162 unit uPSI_JvSimpleXml;                  //JCL
21113: 163 unit uPSI_JclSimpleXml;                 //JCL
21114: 164 unit uPSI_JvXmlDatabase;                //JCL
21115: 165 unit uPSI_JvMaxPixel;                   //JCL
21116: 166 unit uPSI_JvItemsSearchs;               //JCL
21117: 167 unit uPSI_StExpEng2;                    //SysTools4
21118: 168 unit uPSI_StGenLog;                     //SysTools4
21119: 169 unit uPSI_JvLogFile;                    //Jcl
21120: 170 unit uPSI_CPort;                        //ComPort Lib v4.11
21121: 171 unit uPSI_CPortCtl;                     //ComPort
21122: 172 unit uPSI_CPortEsc;                     //ComPort
21123: 173 unit BarCodeScaner;                     //ComPort
21124: 174 unit uPSI_JvGraph;                      //JCL
21125: 175 unit uPSI_JvComCtrls;                   //JCL
21126: 176 unit uPSI_GUITesting;                   //D Unit
21127: 177 unit uPSI_JvFindFiles;                  //JCL
21128: 178 unit uPSI_StSystem;                     //SysTools4
21129: 179 unit uPSI_JvKeyboardStates;             //JCL
21130: 180 unit uPSI_JvMail;                       //JCL
21131: 181 unit uPSI_JclConsole;                   //JCL
21132: 182 unit uPSI_JclLANMan;                    //JCL
21133: 183 unit uPSI_IdCustomHTTPServer;           //Indy
21134: 184 unit IdHTTPServer                       //Indy
21135: 185 unit uPSI_IdTCPServer;                  //Indy
21136: 186 unit uPSI_IdSocketHandle;               //Indy
21137: 187 unit uPSI_IdIOHandlerSocket;            //Indy
21138: 188 unit IdIOHandler;                       //Indy
21139: 189 unit uPSI_cutils;                       //Bloodshed
21140: 190 unit uPSI_BoldUtils;                    //boldsoft
21141: 191 unit uPSI_IdSimpleServer;               //Indy
21142: 192 unit uPSI_IdSSLOpenSSL;                 //Indy
21143: 193 unit uPSI_IdMultipartFormData;          //Indy
21144: 194 unit uPSI_SynURIOpener;                 //SynEdit
21145: 195 unit uPSI_PerlRegEx;                    //PCRE
21146: 196 unit uPSI_IdHeaderList;                 //Indy
21147: 197 unit uPSI_StFirst;                      //SysTools4
21148: 198 unit uPSI_JvCtrls;                      //JCL
21149: 199 unit uPSI_IdTrivialFTPBase;             //Indy
21150: 200 unit uPSI_IdTrivialFTP;                 //Indy
```

```
21151: 201 unit uPSI_IdUDPBase;                      //Indy
21152: 202 unit uPSI_IdUDPClient;                    //Indy
21153: 203 unit uPSI_utypes;                         //for DMath.DLL
21154: 204 unit uPSI_ShellAPI;                       //Borland
21155: 205 unit uPSI_IdRemoteCMDClient;              //Indy
21156: 206 unit uPSI_IdRemoteCMDServer;              //Indy
21157: 207 unit IdRexecServer;                       //Indy
21158: 208 unit IdRexec; (unit uPSI_IdRexec;)        //Indy
21159: 209 unit IdUDPServer;                         //Indy
21160: 210 unit IdTimeUDPServer;                     //Indy
21161: 211 unit IdTimeServer;                        //Indy
21162: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)  //Indy
21163: 213 unit uPSI_IdIPWatch;                      //Indy
21164: 214 unit uPSI_IdIrcServer;                    //Indy
21165: 215 unit uPSI_IdMessageCollection;            //Indy
21166: 216 unit uPSI_cPEM;                           //Fundamentals 4
21167: 217 unit uPSI_cFundamentUtils;                //Fundamentals 4
21168: 218 unit uPSI_uwinplot;                       //DMath
21169: 219 unit uPSI_xrtl_util_CPUUtils;             //ExtentedRTL
21170: 220 unit uPSI_GR32_System;                    //Graphics32
21171: 221 unit uPSI_cFileUtils;                     //Fundamentals 4
21172: 222 unit uPSI_cDateTime; (timemachine)        //Fundamentals 4
21173: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
21174: 224 unit uPSI_cRandom;                        //Fundamentals 4
21175: 225 unit uPSI_ueval;                          //DMath
21176: 226 unit uPSI_xrtl_net_URIUtils;              //ExtendedRTL
21177: 227 unit xrtl_net_URIUtils;                   //ExtendedRTL
21178: 228 unit uPSI_ufft;  (FFT)                    //DMath
21179: 229 unit uPSI_DBXChannel;                     //Delphi
21180: 230 unit uPSI_DBXIndyChannel;                 //Delphi Indy
21181: 231 unit uPSI_xrtl_util_COMCat;               //ExtendedRTL
21182: 232 unit uPSI_xrtl_util_StrUtils;             //ExtendedRTL
21183: 233 unit uPSI_xrtl_util_VariantUtils;         //ExtendedRTL
21184: 234 unit uPSI_xrtl_util_FileUtils;            //ExtendedRTL
21185: 235 unit xrtl_util_Compat;                    //ExtendedRTL
21186: 236 unit uPSI_OleAuto;                        //Borland
21187: 237 unit uPSI_xrtl_util_COMUtils;             //ExtendedRTL
21188: 238 unit uPSI_CmAdmCtl;                       //Borland
21189: 239 unit uPSI_ValEdit2;                       //VCL
21190: 240 unit uPSI_GR32;  //Graphics32            //Graphics32
21191: 241 unit uPSI_GR32_Image;                     //Graphics32
21192: 242 unit uPSI_xrtl_util_TimeUtils;            //ExtendedRTL
21193: 243 unit uPSI_xrtl_util_TimeZone;             //ExtendedRTL
21194: 244 unit uPSI_xrtl_util_TimeStamp;            //ExtendedRTL
21195: 245 unit uPSI_xrtl_util_Map;                  //ExtendedRTL
21196: 246 unit uPSI_xrtl_util_Set;                  //ExtendedRTL
21197: 247 unit uPSI_CPortMonitor;                   //ComPort
21198: 248 unit uPSI_StIniStm;                       //SysTools4
21199: 249 unit uPSI_GR32_ExtImage;                  //Graphics32
21200: 250 unit uPSI_GR32_OrdinalMaps;               //Graphics32
21201: 251 unit uPSI_GR32_Rasterizers;               //Graphics32
21202: 252 unit uPSI_xrtl_util_Exception;            //ExtendedRTL
21203: 253 unit uPSI_xrtl_util_Value;                //ExtendedRTL
21204: 254 unit uPSI_xrtl_util_Compare;              //ExtendedRTL
21205: 255 unit uPSI_FlatSB;                         //VCL
21206: 256 unit uPSI_JvAnalogClock;                  //JCL
21207: 257 unit uPSI_JvAlarms;                       //JCL
21208: 258 unit uPSI_JvSQLS;                         //JCL
21209: 259 unit uPSI_JvDBSecur;                      //JCL
21210: 260 unit uPSI_JvDBQBE;                        //JCL
21211: 261 unit uPSI_JvStarfield;                    //JCL
21212: 262 unit uPSI_JVCLMiscal;                     //JCL
21213: 263 unit uPSI_JvProfiler32;                   //JCL
21214: 264 unit uPSI_JvDirectories,                  //JCL
21215: 265 unit uPSI_JclSchedule,                    //JCL
21216: 266 unit uPSI_JclSvcCtrl,                     //JCL
21217: 267 unit uPSI_JvSoundControl,                 //JCL
21218: 268 unit uPSI_JvBDESQLScript,                 //JCL
21219: 269 unit uPSI_JvgDigits,                      //JCL>
21220: 270 unit uPSI_ImgList;                        //TCustomImageList
21221: 271 unit uPSI_JclMIDI;                        //JCL>
21222: 272 unit uPSI_JclWinMidi;                     //JCL>
21223: 273 unit uPSI_JclNTFS;                        //JCL>
21224: 274 unit uPSI_JclAppInst;                     //JCL>
21225: 275 unit uPSI_JvRle;                          //JCL>
21226: 276 unit uPSI_JvRas32;                        //JCL>
21227: 277 unit uPSI_JvImageDrawThread,              //JCL>
21228: 278 unit uPSI_JvImageWindow,                  //JCL>
21229: 279 unit uPSI_JvTransparentForm;              //JCL>
21230: 280 unit uPSI_JvWinDialogs;                   //JCL>
21231: 281 unit uPSI_JvSimLogic,                     //JCL>
21232: 282 unit uPSI_JvSimIndicator,                 //JCL>
21233: 283 unit uPSI_JvSimPID,                       //JCL>
21234: 284 unit uPSI_JvSimPIDLinker,                 //JCL>
21235: 285 unit uPSI_IdRFCReply;                     //Indy
21236: 286 unit uPSI_IdIdent;                        //Indy
21237: 287 unit uPSI_IdIdentServer;                  //Indy
21238: 288 unit uPSI_JvPatchFile;                    //JCL
21239: 289 unit uPSI_StNetPfm;                       //SysTools4
```

```
21240: 290 unit uPSI_StNet;                          //SysTools4
21241: 291 unit uPSI_JclPeImage;                     //JCL
21242: 292 unit uPSI_JclPrint;                       //JCL
21243: 293 unit uPSI_JclMime;                       //JCL
21244: 294 unit uPSI_JvRichEdit;                     //JCL
21245: 295 unit uPSI_JvDBRichEd;                     //JCL
21246: 296 unit uPSI_JvDice;                        //JCL
21247: 297 unit uPSI_JvFloatEdit;                   //JCL 3.9.8
21248: 298 unit uPSI_JvDirFrm;                      //JCL
21249: 299 unit uPSI_JvDualList;                    //JCL
21250: 300 unit uPSI_JvSwitch;                      ////JCL
21251: 301 unit uPSI_JvTimerLst;                    ////JCL
21252: 302 unit uPSI_JvMemTable;                    //JCL
21253: 303 unit uPSI_JvObjStr;                      //JCL
21254: 304 unit uPSI_StLArr;                        //SysTools4
21255: 305 unit uPSI_StWmDCpy;                      //SysTools4
21256: 306 unit uPSI_StText;                        //SysTools4
21257: 307 unit uPSI_StNTLog;                       //SysTools4
21258: 308 unit uPSI_xrtl_math_Integer;             //ExtendedRTL
21259: 309 unit uPSI_JvImagPrvw;                    //JCL
21260: 310 unit uPSI_JvFormPatch;                   //JCL
21261: 311 unit uPSI_JvPicClip;                     //JCL
21262: 312 unit uPSI_JvDataConv;                    //JCL
21263: 313 unit uPSI_JvCpuUsage;                    //JCL
21264: 314 unit uPSI_JclUnitConv_mX2;               //JCL
21265: 315 unit JvDualListForm;                     //JCL
21266: 316 unit uPSI_JvCpuUsage2;                   //JCL
21267: 317 unit uPSI_JvParserForm;                  //JCL
21268: 318 unit uPSI_JvJanTreeView;                 //JCL
21269: 319 unit uPSI_JvTransLED;                    //JCL
21270: 320 unit uPSI_JvPlaylist;                    //JCL
21271: 321 unit uPSI_JvFormAutoSize;                //JCL
21272: 322 unit uPSI_JvYearGridEditForm;            //JCL
21273: 323 unit uPSI_JvMarkupCommon;                //JCL
21274: 324 unit uPSI_JvChart;                       //JCL
21275: 325 unit uPSI_JvXPCore;                      //JCL
21276: 326 unit uPSI_JvXPCoreUtils;                 //JCL
21277: 327 unit uPSI_StatsClasses;                  //mX4
21278: 328 unit uPSI_ExtCtrls2;                     //VCL
21279: 329 unit uPSI_JvUrlGrabbers;                 //JCL
21280: 330 unit uPSI_JvXmlTree;                     //JCL
21281: 331 unit uPSI_JvWavePlayer;                  //JCL
21282: 332 unit uPSI_JvUnicodeCanvas;               //JCL
21283: 333 unit uPSI_JvTFUtils;                     //JCL
21284: 334 unit uPSI_IdServerIOHandler;             //Indy
21285: 335 unit uPSI_IdServerIOHandlerSocket;       //Indy
21286: 336 unit uPSI_IdMessageCoder;                //Indy
21287: 337 unit uPSI_IdMessageCoderMIME;            //Indy
21288: 338 unit uPSI_IdMIMETypes;                   //Indy
21289: 339 unit uPSI_JvConverter;                   //JCL
21290: 340 unit uPSI_JvCsvParse;                    //JCL
21291: 341 unit uPSI_umath;  unit uPSI_ugamma;      //DMath
21292: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
21293: 343 unit uPSI_JvDBGridExport;                //JCL
21294: 344 unit uPSI_JvgExport;                     //JCL
21295: 345 unit uPSI_JvSerialMaker;                 //JCL
21296: 346 unit uPSI_JvWin32;                       //JCL
21297: 347 unit uPSI_JvPaintFX;                     //JCL
21298: 348 unit uPSI_JvOracleDataSet; (beta)        //JCL
21299: 349 unit uPSI_JvValidators; (preview)        //JCL
21300: 350 unit uPSI_JvNTEventLog;                  //JCL
21301: 351 unit uPSI_ShellZipTool;                  //mX4
21302: 352 unit uPSI_JvJoystick;                    //JCL
21303: 353 unit uPSI_JvMailSlots;                   //JCL
21304: 354 unit uPSI_JclComplex;                    //JCL
21305: 355 unit uPSI_SynPdf;                        //Synopse
21306: 356 unit uPSI_Registry;                      //VCL
21307: 357 unit uPSI_TlHelp32;                      //VCL
21308: 358 unit uPSI_JclRegistry;                   //JCL
21309: 359 unit uPSI_JvAirBrush;                    //JCL
21310: 360 unit uPSI_mORMotReport;                  //Synopse
21311: 361 unit uPSI_JclLocales;                    //JCL
21312: 362 unit uPSI_SynEdit;                       //SynEdit
21313: 363 unit uPSI_SynEditTypes;                  //SynEdit
21314: 364 unit uPSI_SynMacroRecorder;              //SynEdit
21315: 365 unit uPSI_LongIntList;                   //SynEdit
21316: 366 unit uPSI_devcutils;                     //DevC
21317: 367 unit uPSI_SynEditMiscClasses;            //SynEdit
21318: 368 unit uPSI_SynEditRegexSearch;            //SynEdit
21319: 369 unit uPSI_SynEditHighlighter;            //SynEdit
21320: 370 unit uPSI_SynHighlighterPas;             //SynEdit
21321: 371 unit uPSI_JvSearchFiles;                 //JCL
21322: 372 unit uPSI_SynHighlighterAny;             //Lazarus
21323: 373 unit uPSI_SynEditKeyCmds;                //SynEdit
21324: 374 unit uPSI_SynEditMiscProcs,              //SynEdit
21325: 375 unit uPSI_SynEditKbdHandler              //SynEdit
21326: 376 unit uPSI_JvAppInst,                     //JCL
21327: 377 unit uPSI_JvAppEvent;                    //JCL
21328: 378 unit uPSI_JvAppCommand;                  //JCL
```

```
21329: 379 unit uPSI_JvAnimTitle;                    //JCL
21330: 380 unit uPSI_JvAnimatedImage;                //JCL
21331: 381 unit uPSI_SynEditExport;                  //SynEdit
21332: 382 unit uPSI_SynExportHTML;                  //SynEdit
21333: 383 unit uPSI_SynExportRTF;                   //SynEdit
21334: 384 unit uPSI_SynEditSearch;                  //SynEdit
21335: 385 unit uPSI_fMain_back                      //maXbox;
21336: 386 unit uPSI_JvZoom;                         //JCL
21337: 387 unit uPSI_PMrand;                         //PM
21338: 388 unit uPSI_JvSticker;                      //JCL
21339: 389 unit uPSI_XmlVerySimple;                  //mX4
21340: 390 unit uPSI_Services;                       //ExtPascal
21341: 391 unit uPSI_ExtPascalUtils;                 //ExtPascal
21342: 392 unit uPSI_SocketsDelphi;                  //ExtPascal
21343: 393 unit uPSI_StBarC;                         //SysTools
21344: 394 unit uPSI_StDbBarC;                       //SysTools
21345: 395 unit uPSI_StBarPN;                        //SysTools
21346: 396 unit uPSI_StDbPNBC;                       //SysTools
21347: 397 unit uPSI_StDb2DBC;                       //SysTools
21348: 398 unit uPSI_StMoney;                        //SysTools
21349: 399 unit uPSI_JvForth;                        //JCL
21350: 400 unit uPSI_RestRequest;                    //mX4
21351: 401 unit uPSI_HttpRESTConnectionIndy;         //mX4
21352: 402 unit uPSI_JvXmlDatabase; //update        //JCL
21353: 403 unit uPSI_StAstro;                        //SysTools
21354: 404 unit uPSI_StSort;                         //SysTools
21355: 405 unit uPSI_StDate;                         //SysTools
21356: 406 unit uPSI_StDateSt;                       //SysTools
21357: 407 unit uPSI_StBase;                         //SysTools
21358: 408 unit uPSI_StVInfo;                        //SysTools
21359: 409 unit uPSI_JvBrowseFolder;                 //JCL
21360: 410 unit uPSI_JvBoxProcs;                     //JCL
21361: 411 unit uPSI_urandom; (unit uranuvag;)       //DMath
21362: 412 unit uPSI_usimann; (unit ugenalg;)        //DMath
21363: 413 unit uPSI_JvHighlighter;                  //JCL
21364: 414 unit uPSI_Diff;                           //mX4
21365: 415 unit uPSI_SpringWinAPI;                   //DSpring
21366: 416 unit uPSI_StBits;                         //SysTools
21367: 417 unit uPSI_TomDBQue;                       //mX4
21368: 418 unit uPSI_MultilangTranslator;            //mX4
21369: 419 unit uPSI_HyperLabel;                     //mX4
21370: 420 unit uPSI_Starter;                        //mX4
21371: 421 unit uPSI_FileAssocs;                     //devC
21372: 422 unit uPSI_devFileMonitorX;                //devC
21373: 423 unit uPSI_devrun;                         //devC
21374: 424 unit uPSI_devExec;                        //devC
21375: 425 unit uPSI_oysUtils;                       //devC
21376: 426 unit uPSI_DosCommand;                     //devC
21377: 427 unit uPSI_CppTokenizer;                   //devC
21378: 428 unit uPSI_JvHLParser;                     //devC
21379: 429 unit uPSI_JclMapi;                        //JCL
21380: 430 unit uPSI_JclShell;                       //JCL
21381: 431 unit uPSI_JclCOM;                         //JCL
21382: 432 unit uPSI_GR32_Math;                      //Graphics32
21383: 433 unit uPSI_GR32_LowLevel;                  //Graphics32
21384: 434 unit uPSI_SimpleHl;                       //mX4
21385: 435 unit uPSI_GR32_Filters,                   //Graphics32
21386: 436 unit uPSI_GR32_VectorMaps;                //Graphics32
21387: 437 unit uPSI_cXMLFunctions;                  //Fundamentals 4
21388: 438 unit uPSI_JvTimer;                        //JCL
21389: 439 unit uPSI_cHTTPUtils;                     //Fundamentals 4
21390: 440 unit uPSI_cTLSUtils;                      //Fundamentals 4
21391: 441 unit uPSI_JclGraphics;                    //JCL
21392: 442 unit uPSI_JclSynch;                       //JCL
21393: 443 unit uPSI_IdTelnet;                       //Indy
21394: 444 unit uPSI_IdTelnetServer,                 //Indy
21395: 445 unit uPSI_IdEcho,                         //Indy
21396: 446 unit uPSI_IdEchoServer,                   //Indy
21397: 447 unit uPSI_IdEchoUDP,                      //Indy
21398: 448 unit uPSI_IdEchoUDPServer,                //Indy
21399: 449 unit uPSI_IdSocks,                        //Indy
21400: 450 unit uPSI_IdAntiFreezeBase;               //Indy
21401: 451 unit uPSI_IdHostnameServer;               //Indy
21402: 452 unit uPSI_IdTunnelCommon,                 //Indy
21403: 453 unit uPSI_IdTunnelMaster,                 //Indy
21404: 454 unit uPSI_IdTunnelSlave,                  //Indy
21405: 455 unit uPSI_IdRSH,                          //Indy
21406: 456 unit uPSI_IdRSHServer,                    //Indy
21407: 457 unit uPSI_Spring_Cryptography_Utils;      //Spring4Delphi
21408: 458 unit uPSI_MapReader,                      //devC
21409: 459 unit uPSI_LibTar,                         //devC
21410: 460 unit uPSI_IdStack;                        //Indy
21411: 461 unit uPSI_IdBlockCipherIntercept;         //Indy
21412: 462 unit uPSI_IdChargenServer;                //Indy
21413: 463 unit uPSI_IdFTPServer,                    //Indy
21414: 464 unit uPSI_IdException,                    //Indy
21415: 465 unit uPSI_utexplot;                       //DMath
21416: 466 unit uPSI_uwinstr;                        //DMath
21417: 467 unit uPSI_VarRecUtils;                    //devC
```

```
21418: 468 unit uPSI_JvStringListToHtml,              //JCL
21419: 469 unit uPSI_JvStringHolder,                  //JCL
21420: 470 unit uPSI_IdCoder;                         //Indy
21421: 471 unit uPSI_SynHighlighterDfm;               //Synedit
21422: 472 unit uHighlighterProcs; in 471             //Synedit
21423: 473 unit uPSI_LazFileUtils,                    //LCL
21424: 474 unit uPSI_IDECmdLine;                      //LCL
21425: 475 unit uPSI_lazMasks;                        //LCL
21426: 476 unit uPSI_ip_misc;                         //mX4
21427: 477 unit uPSI_Barcode;                         //LCL
21428: 478 unit uPSI_SimpleXML;                       //LCL
21429: 479 unit uPSI_JclIniFiles;                     //JCL
21430: 480 unit uPSI_D2XXUnit;  {$X-}                 //FTDI
21431: 481 unit uPSI_JclDateTime;                     //JCL
21432: 482 unit uPSI_JclEDI;                          //JCL
21433: 483 unit uPSI_JclMiscel2;                      //JCL
21434: 484 unit uPSI_JclValidation;                   //JCL
21435: 485 unit uPSI_JclAnsiStrings; {-PString}      //JCL
21436: 486 unit uPSI_SynEditMiscProcs2;               //Synedit
21437: 487 unit uPSI_JclStreams;                      //JCL
21438: 488 unit uPSI_QRCode;                          //mX4
21439: 489 unit uPSI_BlockSocket;                     //ExtPascal
21440: 490 unit uPSI_Masks,Utils                      //VCL
21441: 491 unit uPSI_synautil;                        //Synapse!
21442: 492 unit uPSI_JclMath_Class;                   //JCL RTL
21443: 493 unit ugamdist; //Gamma function           //DMath
21444: 494 unit uibeta, ucorrel; //IBeta             //DMath
21445: 495 unit uPSI_SRMgr;                           //mX4
21446: 496 unit uPSI_HotLog;                          //mX4
21447: 497 unit uPSI_DebugBox;                        //mX4
21448: 498 unit uPSI_ustrings;                        //DMath
21449: 499 unit uPSI_uregtest;                        //DMath
21450: 500 unit uPSI_usimplex;                        //DMath
21451: 501 unit uPSI_uhyper;                          //DMath
21452: 502 unit uPSI_IdHL7;                           //Indy
21453: 503 unit uPSI_IdIPMCastBase,                   //Indy
21454: 504 unit uPSI_IdIPMCastServer;                 //Indy
21455: 505 unit uPSI_IdIPMCastClient;                 //Indy
21456: 506 unit uPSI_unlfit; //nlregression          //DMath
21457: 507 unit uPSI_IdRawHeaders;                    //Indy
21458: 508 unit uPSI_IdRawClient;                     //Indy
21459: 509 unit uPSI_IdRawFunctions;                  //Indy
21460: 510 unit uPSI_IdTCPStream;                     //Indy
21461: 511 unit uPSI_IdSNPP;                          //Indy
21462: 512 unit uPSI_St2DBarC;                        //SysTools
21463: 513 unit uPSI_ImageWin;  //FTL                //VCL
21464: 514 unit uPSI_CustomDrawTreeView; //FTL       //VCL
21465: 515 unit uPSI_GraphWin;  //FTL                //VCL
21466: 516 unit uPSI_actionMain;  //FTL              //VCL
21467: 517 unit uPSI_StSpawn;                         //SysTools
21468: 518 unit uPSI_CtlPanel;                        //VCL
21469: 519 unit uPSI_IdLPR;                           //Indy
21470: 520 unit uPSI_SockRequestInterpreter;          //Indy
21471: 521 unit uPSI_ulambert;                        //DMath
21472: 522 unit uPSI_ucholesk;                        //DMath
21473: 523 unit uPSI_SimpleDS;                        //VCL
21474: 524 unit uPSI_DBXSqlScanner;                   //VCL
21475: 525 unit uPSI_DBXMetaDataUtil;                 //VCL
21476: 526 unit uPSI_Chart;                           //TEE
21477: 527 unit uPSI_TeeProcs;                        //TEE
21478: 528 unit mXBDEUtils;                           //mX4
21479: 529 unit uPSI_MDIEdit;                         //VCL
21480: 530 unit uPSI_CopyPrsr;                        //VCL
21481: 531 unit uPSI_SockApp;                         //VCL
21482: 532 unit uPSI_AppEvnts;                        //VCL
21483: 533 unit uPSI_ExtActns;                        //VCL
21484: 534 unit uPSI_TeEngine;                        //TEE
21485: 535 unit uPSI_CoolMain; //browser             //VCL
21486: 536 unit uPSI_StCRC;                           //SysTools
21487: 537 unit uPSI_StDecMth2;                       //SysTools
21488: 538 unit uPSI_frmExportMain;                   //Synedit
21489: 539 unit uPSI_SynDBEdit;                       //Synedit
21490: 540 unit uPSI_SynEditWildcardSearch;          //Synedit
21491: 541 unit uPSI_BoldComUtils;                    //BOLD
21492: 542 unit uPSI_BoldIsoDateTime;                 //BOLD
21493: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils     //BOLD
21494: 544 unit uPSI_BoldXMLRequests;                 //BOLD
21495: 545 unit uPSI_BoldStringList;                  //BOLD
21496: 546 unit uPSI_BoldFileHandler;                 //BOLD
21497: 547 unit uPSI_BoldContainers;                  //BOLD
21498: 548 unit uPSI_BoldQueryUserDlg;                //BOLD
21499: 549 unit uPSI_BoldWinINet;                     //BOLD
21500: 550 unit uPSI_BoldQueue;                       //BOLD
21501: 551 unit uPSI_JvPcx;                           //JCL
21502: 552 unit uPSI_IdWhois;                         //Indy
21503: 553 unit uPSI_IdWhoIsServer;                   //Indy
21504: 554 unit uPSI_IdGopher;                        //Indy
21505: 555 unit uPSI_IdDateTimeStamp;                 //Indy
21506: 556 unit uPSI_IdDayTimeServer;                 //Indy
```

```
21507: 557 unit uPSI_IdDayTimeUDP;                  //Indy
21508: 558 unit uPSI_IdDayTimeUDPServer;            //Indy
21509: 559 unit uPSI_IdDICTServer;                  //Indy
21510: 560 unit uPSI_IdDiscardServer;               //Indy
21511: 561 unit uPSI_IdDiscardUDPServer;            //Indy
21512: 562 unit uPSI_IdMappedFTP;                   //Indy
21513: 563 unit uPSI_IdMappedPortTCP;               //Indy
21514: 564 unit uPSI_IdGopherServer;                //Indy
21515: 565 unit uPSI_IdQotdServer;                  //Indy
21516: 566 unit uPSI_JvRgbToHtml;                   //JCL
21517: 567 unit uPSI_JvRemLog,                      //JCL
21518: 568 unit uPSI_JvSysComp;                     //JCL
21519: 569 unit uPSI_JvTMTL;                        //JCL
21520: 570 unit uPSI_JvWinampAPI;                   //JCL
21521: 571 unit uPSI_MSysUtils;                     //mX4
21522: 572 unit uPSI_ESBMaths;                      //ESB
21523: 573 unit uPSI_ESBMaths2;                     //ESB
21524: 574 unit uPSI_uLkJSON;                       //Lk
21525: 575 unit uPSI_ZURL;  //Zeos                 //Zeos
21526: 576 unit uPSI_ZSysUtils;                     //Zeos
21527: 577 unit unaUtils internals                  //UNA
21528: 578 unit uPSI_ZMatchPattern;                 //Zeos
21529: 579 unit uPSI_ZClasses;                      //Zeos
21530: 580 unit uPSI_ZCollections;                  //Zeos
21531: 581 unit uPSI_ZEncoding;                     //Zeos
21532: 582 unit uPSI_IdRawBase;                     //Indy
21533: 583 unit uPSI_IdNTLM;                        //Indy
21534: 584 unit uPSI_IdNNTP;                        //Indy
21535: 585 unit uPSI_usniffer; //PortScanForm      //mX4
21536: 586 unit uPSI_IdCoderMIME;                   //Indy
21537: 587 unit uPSI_IdCoderUUE;                    //Indy
21538: 588 unit uPSI_IdCoderXXE;                    //Indy
21539: 589 unit uPSI_IdCoder3to4;                   //Indy
21540: 590 unit uPSI_IdCookie;                      //Indy
21541: 591 unit uPSI_IdCookieManager;               //Indy
21542: 592 unit uPSI_WDosSocketUtils;               //WDos
21543: 593 unit uPSI_WDosPlcUtils;                  //WDos
21544: 594 unit uPSI_WDosPorts;                     //WDos
21545: 595 unit uPSI_WDosResolvers;                 //WDos
21546: 596 unit uPSI_WDosTimers;                    //WDos
21547: 597 unit uPSI_WDosPlcs;                      //WDos
21548: 598 unit uPSI_WDosPneumatics;                //WDos
21549: 599 unit uPSI_IdFingerServer;                //Indy
21550: 600 unit uPSI_IdDNSResolver;                 //Indy
21551: 601 unit uPSI_IdHTTPWebBrokerBridge;         //Indy
21552: 602 unit uPSI_IdIntercept;                   //Indy
21553: 603 unit uPSI_IdIPMCastBase;                 //Indy
21554: 604 unit uPSI_IdLogBase;                     //Indy
21555: 605 unit uPSI_IdIOHandlerStream;             //Indy
21556: 606 unit uPSI_IdMappedPortUDP;               //Indy
21557: 607 unit uPSI_IdQOTDUDPServer;               //Indy
21558: 608 unit uPSI_IdQOTDUDP;                     //Indy
21559: 609 unit uPSI_IdSysLog;                      //Indy
21560: 610 unit uPSI_IdSysLogServer;                //Indy
21561: 611 unit uPSI_IdSysLogMessage;               //Indy
21562: 612 unit uPSI_IdTimeServer;                  //Indy
21563: 613 unit uPSI_IdTimeUDP;                     //Indy
21564: 614 unit uPSI_IdTimeUDPServer;               //Indy
21565: 615 unit uPSI_IdUserAccounts;                //Indy
21566: 616 unit uPSI_TextUtils;                     //mX4
21567: 617 unit uPSI_MandelbrotEngine;              //mX4
21568: 618 unit uPSI_delphi_arduino_Unit1;          //mX4
21569: 619 unit uPSI_DTDSchema2;                    //mX4
21570: 620 unit uPSI_fplotMain;                     //DMath
21571: 621 unit uPSI_FindFileIter;                  //mX4
21572: 622 unit uPSI_PppState;  (JclStrHashMap)     //PPP
21573: 623 unit uPSI_PppParser;                     //PPP
21574: 624 unit uPSI_PppLexer;                      //PPP
21575: 625 unit uPSI_PCharUtils;                    //PPP
21576: 626 unit uPSI_uJSON;                         //WU
21577: 627 unit uPSI_JclStrHashMap;                 //JCL
21578: 628 unit uPSI_JclHookExcept;                 //JCL
21579: 629 unit uPSI_EncdDecd;                      //VCL
21580: 630 unit uPSI_SockAppReg;                    //VCL
21581: 631 unit uPSI_PJFileHandle;                  //PJ
21582: 632 unit uPSI_PJEnvVars;                     //PJ
21583: 633 unit uPSI_PJPipe;                        //PJ
21584: 634 unit uPSI_PJPipeFilters;                 //PJ
21585: 635 unit uPSI_PJConsoleApp;                  //PJ
21586: 636 unit uPSI_UConsoleAppEx;                 //PJ
21587: 637 unit uPSI_DbxSocketChannelNative,        //VCL
21588: 638 unit uPSI_DbxDataGenerator,              //VCL
21589: 639 unit uPSI_DBXClient;                     //VCL
21590: 640 unit uPSI_IdLogEvent;                    //Indy
21591: 641 unit uPSI_Reversi;                       //mX4
21592: 642 unit uPSI_Geometry;                      //mX4
21593: 643 unit uPSI_IdSMTPServer;                  //Indy
21594: 644 unit uPSI_Textures;                      //mX4
21595: 645 unit uPSI_IBX;                           //VCL
```

```
21596: 646 unit uPSI_IWDBCommon;                    //VCL
21597: 647 unit uPSI_SortGrid;                      //mX4
21598: 648 unit uPSI_IB;                           //VCL
21599: 649 unit uPSI_IBScript;                     //VCL
21600: 650 unit uPSI_JvCSVBaseControls;            //JCL
21601: 651 unit uPSI_Jvg3DColors;                  //JCL
21602: 652 unit uPSI_JvHLEditor;  //beat           //JCL
21603: 653 unit uPSI_JvShellHook;                  //JCL
21604: 654 unit uPSI_DBCommon2                     //VCL
21605: 655 unit uPSI_JvSHFileOperation;            //JCL
21606: 656 unit uPSI_uFilexport;                   //mX4
21607: 657 unit uPSI_JvDialogs;                    //JCL
21608: 658 unit uPSI_JvDBTreeView;                 //JCL
21609: 659 unit uPSI_JvDBUltimGrid;                //JCL
21610: 660 unit uPSI_JvDBQueryParamsForm;          //JCL
21611: 661 unit uPSI_JvExControls;                 //JCL
21612: 662 unit uPSI_JvBDEMemTable;                //JCL
21613: 663 unit uPSI_JvCommStatus;                 //JCL
21614: 664 unit uPSI_JvMailSlots2;                 //JCL
21615: 665 unit uPSI_JvgWinMask;                   //JCL
21616: 666 unit uPSI_StEclpse;                     //SysTools
21617: 667 unit uPSI_StMime;                       //SysTools
21618: 668 unit uPSI_StList;                       //SysTools
21619: 669 unit uPSI_StMerge;                      //SysTools
21620: 670 unit uPSI_StStrS;                       //SysTools
21621: 671 unit uPSI_StTree,                       //SysTools
21622: 672 unit uPSI_StVArr;                       //SysTools
21623: 673 unit uPSI_StRegIni;                     //SysTools
21624: 674 unit uPSI_urkf;                         //DMath
21625: 675 unit uPSI_usvd;                         //DMath
21626: 676 unit uPSI_DepWalkUtils;                 //JCL
21627: 677 unit uPSI_OptionsFrm;                   //JCL
21628: 678 unit yuvconverts;                       //mX4
21629: 679 uPSI_JvPropAutoSave;                    //JCL
21630: 680 uPSI_AclAPI;                            //alcinoe
21631: 681 uPSI_AviCap;                            //alcinoe
21632: 682 uPSI_ALAVLBinaryTree;                   //alcinoe
21633: 683 uPSI_ALFcnMisc;                         //alcinoe
21634: 684 uPSI_ALStringList;                      //alcinoe
21635: 685 uPSI_ALQuickSortList;                   //alcinoe
21636: 686 uPSI_ALStaticText;                      //alcinoe
21637: 687 uPSI_ALJSONDoc;                         //alcinoe
21638: 688 uPSI_ALGSMComm;                         //alcinoe
21639: 689 uPSI_ALWindows;                         //alcinoe
21640: 690 uPSI_ALMultiPartFormDataParser;         //alcinoe
21641: 691 uPSI_ALHttpCommon;                      //alcinoe
21642: 692 uPSI_ALWebSpider,                       //alcinoe
21643: 693 uPSI_ALHttpClient;                      //alcinoe
21644: 694 uPSI_ALFcnHTML;                         //alcinoe
21645: 695 uPSI_ALFTPClient;                       //alcinoe
21646: 696 uPSI_ALInternetMessageCommon;           //alcinoe
21647: 697 uPSI_ALWininetHttpClient;               //alcinoe
21648: 698 uPSI_ALWinInetFTPClient;                //alcinoe
21649: 699 uPSI_ALWinHttpWrapper;                  //alcinoe
21650: 700 uPSI_ALWinHttpClient;                   //alcinoe
21651: 701 uPSI_ALFcnWinSock;                      //alcinoe
21652: 702 uPSI_ALFcnSQL;                          //alcinoe
21653: 703 uPSI_ALFcnCGI;                          //alcinoe
21654: 704 uPSI_ALFcnExecute;                      //alcinoe
21655: 705 uPSI_ALFcnFile;                         //alcinoe
21656: 706 uPSI_ALFcnMime;                         //alcinoe
21657: 707 uPSI_ALPhpRunner;                       //alcinoe
21658: 708 uPSI_ALGraphic;                         //alcinoe
21659: 709 uPSI_ALIniFiles;                        //alcinoe
21660: 710 uPSI_ALMemCachedClient;                 //alcinoe
21661: 711 unit uPSI_MyGrids;                      //mX4
21662: 712 uPSI_ALMultiPartMixedParser             //alcinoe
21663: 713 uPSI_ALSMTPClient                       //alcinoe
21664: 714 uPSI_ALNNTPClient;                      //alcinoe
21665: 715 uPSI_ALHintBalloon;                     //alcinoe
21666: 716 unit uPSI_ALXmlDoc;                     //alcinoe
21667: 717 unit uPSI_IPCThrd;                      //VCL
21668: 718 unit uPSI_MonForm;                      //VCL
21669: 719 unit uPSI_TeCanvas;                     //Orpheus
21670: 720 unit uPSI_Ovcmisc;                      //Orpheus
21671: 721 unit uPSI_ovcfiler;                     //Orpheus
21672: 722 unit uPSI_ovcstate;                     //Orpheus
21673: 723 unit uPSI_ovccoco;                      //Orpheus
21674: 724 unit uPSI_ovcrvexp;                     //Orpheus
21675: 725 unit uPSI_OvcFormatSettings;            //Orpheus
21676: 726 unit uPSI_OvcUtils;                     //Orpheus
21677: 727 unit uPSI_ovcstore;                     //Orpheus
21678: 728 unit uPSI_ovcstr;                       //Orpheus
21679: 729 unit uPSI_ovcmru;                       //Orpheus
21680: 730 unit uPSI_ovccmd;                       //Orpheus
21681: 731 unit uPSI_ovctimer;                     //Orpheus
21682: 732 unit uPSI_ovcintl;                      //Orpheus
21683: 733 uPSI_AfCircularBuffer;                  //AsyncFree
21684: 734 uPSI_AfUtils;                           //AsyncFree
```

```
21685: 735 uPSI_AfSafeSync;                          //AsyncFree
21686: 736 uPSI_AfComPortCore;                       //AsyncFree
21687: 737 uPSI_AfComPort;                           //AsyncFree
21688: 738 uPSI_AfPortControls;                      //AsyncFree
21689: 739 uPSI_AfDataDispatcher;                    //AsyncFree
21690: 740 uPSI_AfViewers;                           //AsyncFree
21691: 741 uPSI_AfDataTerminal;                      //AsyncFree
21692: 742 uPSI_SimplePortMain;                      //AsyncFree
21693: 743 unit uPSI_ovcclock;                       //Orpheus
21694: 744 unit uPSI_o32intlst;                      //Orpheus
21695: 745 unit uPSI_o32ledlabel;                    //Orpheus
21696: 746 unit uPSI_AlMySqlClient;                  //alcinoe
21697: 747 unit uPSI_ALFBXClient;                    //alcinoe
21698: 748 unit uPSI_ALFcnSQL;                       //alcinoe
21699: 749 unit uPSI_AsyncTimer;                     //mX4
21700: 750 unit uPSI_ApplicationFileIO;              //mX4
21701: 751 unit uPSI_PsAPI;                          //VCLé
21702: 752 uPSI_ovcuser;                             //Orpheus
21703: 753 uPSI_ovcurl;                              //Orpheus
21704: 754 uPSI_ovcvlb;                              //Orpheus
21705: 755 uPSI_ovccolor;                            //Orpheus
21706: 756 uPSI_ALFBXLib,                            //alcinoe
21707: 757 uPSI_ovcmeter;                            //Orpheus
21708: 758 uPSI_ovcpeakm;                            //Orpheus
21709: 759 uPSI_O32BGSty;                            //Orpheus
21710: 760 uPSI_ovcBidi;                             //Orpheus
21711: 761 uPSI_ovctcary;                            //Orpheus
21712: 762 uPSI_DXPUtils;                            //mX4
21713: 763 uPSI_ALMultiPartBaseParser;               //alcinoe
21714: 764 uPSI_ALMultiPartAlternativeParser;        //alcinoe
21715: 765 uPSI_ALPOP3Client;                        //alcinoe
21716: 766 uPSI_SmallUtils;                          //mX4
21717: 767 uPSI_MakeApp;                             //mX4
21718: 768 uPSI_O32MouseMon;                         //Orpheus
21719: 769 uPSI_OvcCache;                            //Orpheus
21720: 770 uPSI_ovccalc;                             //Orpheus
21721: 771 uPSI_Joystick,                            //OpenGL
21722: 772 uPSI_ScreenSaver;                         //OpenGL
21723: 773 uPSI_XCollection,                         //OpenGL
21724: 774 uPSI_Polynomials,                         //OpenGL
21725: 775 uPSI_PersistentClasses, //9.86           //OpenGL
21726: 776 uPSI_VectorLists;                         //OpenGL
21727: 777 uPSI_XOpenGL,                            //OpenGL
21728: 778 uPSI_MeshUtils;                           //OpenGL
21729: 779 unit uPSI_JclSysUtils;                    //JCL
21730: 780 unit uPSI_JclBorlandTools;                //JCL
21731: 781 unit JclFileUtils_max;                    //JCL
21732: 782 uPSI_AfDataControls,                      //AsyncFree
21733: 783 uPSI_GLSilhouette;                        //OpenGL
21734: 784 uPSI_JclSysUtils_class;                   //JCL
21735: 785 uPSI_JclFileUtils_class;                  //JCL
21736: 786 uPSI_FileUtil;                            //JCL
21737: 787 uPSI_changefind;                          //mX4
21738: 788 uPSI_cmdIntf;                             //mX4
21739: 789 uPSI_fservice;                            //mX4
21740: 790 uPSI_Keyboard;                            //OpenGL
21741: 791 uPSI_VRMLParser,                          //OpenGL
21742: 792 uPSI_GLFileVRML,                          //OpenGL
21743: 793 uPSI_Octree;                              //OpenGL
21744: 794 uPSI_GLPolyhedron,                        //OpenGL
21745: 795 uPSI_GLCrossPlatform;                     //OpenGL
21746: 796 uPSI_GLParticles;                         //OpenGL
21747: 797 uPSI_GLNavigator;                         //OpenGL
21748: 798 uPSI_GLStarRecord;                        //OpenGL
21749: 799 uPSI_GLTextureCombiners;                  //OpenGL
21750: 800 uPSI_GLCanvas;                            //OpenGL
21751: 801 uPSI_GeometryBB;                          //OpenGL
21752: 802 uPSI_GeometryCoordinates;                 //OpenGL
21753: 803 uPSI_VectorGeometry;                      //OpenGL
21754: 804 uPSI_BumpMapping;                         //OpenGL
21755: 805 uPSI_TGA;                                 //OpenGL
21756: 806 uPSI_GLVectorFileObjects;                 //OpenGL
21757: 807 uPSI_IMM;                                 //VCL
21758: 808 uPSI_CategoryButtons;                     //VCL
21759: 809 uPSI_ButtonGroup;                         //VCL
21760: 810 uPSI_DbExcept;                            //VCL
21761: 811 uPSI_AxCtrls;                             //VCL
21762: 812 uPSI_GL_actorUnit1;                       //OpenGL
21763: 813 uPSI_StdVCL;                              //VCL
21764: 814 unit CurvesAndSurfaces;                   //OpenGL
21765: 815 uPSI_DataAwareMain;                       //AsyncFree
21766: 816 uPSI_TabNotBk;                            //VCL
21767: 817 uPSI_udwsfiler;                           //mX4
21768: 818 uPSI_synaip;                              //Synapse!
21769: 819 uPSI_synacode;                            //Synapse
21770: 820 uPSI_synachar;                            //Synapse
21771: 821 uPSI_synamisc;                            //Synapse
21772: 822 uPSI_synaser;                             //Synapse
21773: 823 uPSI_synaicnv;                            //Synapse
```

```
21774: 824 uPSI_tlntsend;                          //Synapse
21775: 825 uPSI_pingsend;                          //Synapse
21776: 826 uPSI_blcksock;                          //Synapse
21777: 827 uPSI_asn1util;                         //Synapse
21778: 828 uPSI_dnssend;                          //Synapse
21779: 829 uPSI_clamsend;                         //Synapse
21780: 830 uPSI_ldapsend;                         //Synapse
21781: 831 uPSI_mimemess;                         //Synapse
21782: 832 uPSI_slogsend;                         //Synapse
21783: 833 uPSI_mimepart;                         //Synapse
21784: 834 uPSI_mimeinln;                         //Synapse
21785: 835 uPSI_ftpsend,                          //Synapse
21786: 836 uPSI_ftptsend;                         //Synapse
21787: 837 uPSI_httpsend;                         //Synapse
21788: 838 uPSI_sntpsend;                         //Synapse
21789: 839 uPSI_smtpsend;                         //Synapse
21790: 840 uPSI_snmpsend;                         //Synapse
21791: 841 uPSI_imapsend;                         //Synapse
21792: 842 uPSI_pop3send;                         //Synapse
21793: 843 uPSI_nntpsend;                         //Synapse
21794: 844 uPSI_ssl_cryptlib;                     //Synapse
21795: 845 uPSI_ssl_openssl;                      //Synapse
21796: 846 uPSI_synhttp_daemon;                   //Synapse
21797: 847 uPSI_NetWork;                          //mX4
21798: 848 uPSI_PingThread;                       //Synapse
21799: 849 uPSI_JvThreadTimer;                    //JCL
21800: 850 unit uPSI_wwSystem;                    //InfoPower
21801: 851 unit uPSI_IdComponent;                 //Indy
21802: 852 unit uPSI_IdIOHandlerThrottle;         //Indy
21803: 853 unit uPSI_Themes;                      //VCL
21804: 854 unit uPSI_StdStyleActnCtrls;           //VCL
21805: 855 unit uPSI_UDDIHelper;                  //VCL
21806: 856 unit uPSI_IdIMAP4Server;               //Indy
21807: 857 uPSI_VariantSymbolTable,               //VCL //3.9.9.92
21808: 858 uPSI_udf_glob,                         //mX4
21809: 859 uPSI_TabGrid,                          //VCL
21810: 860 uPSI_JsDBTreeView,                     //mX4
21811: 861 uPSI_JsSendMail,                       //mX4
21812: 862 uPSI_dbTvRecordList,                   //mX4
21813: 863 uPSI_TreeVwEx,                        //mX4
21814: 864 uPSI_ECDataLink,                      //mX4
21815: 865 uPSI_dbTree,                          //mX4
21816: 866 uPSI_dbTreeCBox,                      //mX4
21817: 867 unit uPSI_Debug; //TfrmDebug         //mX4
21818: 868 uPSI_TimeFncs;                        //mX4
21819: 869 uPSI_FileIntf,                        //VCL
21820: 870 uPSI_SockTransport,                   //RTL
21821: 871 unit uPSI_WinInet;                    //RTL
21822: 872 unit uPSI_Wwstr;                      //mX4
21823: 873 uPSI_DBLookup,                        //VCL
21824: 874 uPSI_Hotspot,                         //mX4
21825: 875 uPSI_HList; //History List           //mX4
21826: 876 unit uPSI_DrTable;                    //VCL
21827: 877 uPSI_TConnect,                        //VCL
21828: 878 uPSI_DataBkr,                         //VCL
21829: 879 uPSI_HTTPIntr;                        //VCL
21830: 880 unit uPSI_Mathbox;                    //mX4
21831: 881 uPSI_cyIndy,                          //cY
21832: 882 uPSI_cySysUtils,                      //cY
21833: 883 uPSI_cyWinUtils,                      //cY
21834: 884 uPSI_cyStrUtils,                      //cY
21835: 885 uPSI_cyObjUtils,                      //cY
21836: 886 uPSI_cyDateUtils,                     //cY
21837: 887 uPSI_cyBDE,                           //cY
21838: 888 uPSI_cyClasses,                       //cY
21839: 889 uPSI_cyGraphics,  //3.9.9.94_2       //cY
21840: 890 unit uPSI_cyTypes;                    //cY
21841: 891 uPSI_JvDateTimePicker,                //JCL
21842: 892 uPSI_JvCreateProcess,                 //JCL
21843: 893 uPSI_JvEasterEgg,                     //JCL
21844: 894 uPSI_WinSvc,  //3.9.9.94_3           //VCL
21845: 895 uPSI_SvcMgr                           //VCL
21846: 896 unit uPSI_JvPickDate;                 //JCL
21847: 897 unit uPSI_JvNotify;                   //JCL
21848: 898 uPSI_JvStrHlder                       //JCL
21849: 899 unit uPSI_JclNTFS2;                   //JCL
21850: 900 uPSI_Jcl8087 //math coprocessor       //JCL
21851: 901 uPSI_JvAddPrinter                     //JCL
21852: 902 uPSI_JvCabFile                        //JCL
21853: 903 uPSI_JvDataEmbedded;                  //JCL
21854: 904 unit uPSI_U_HexView;                  //mX4
21855: 905 uPSI_UWavein4,                        //mX4
21856: 906 uPSI_AMixer,                          //mX4
21857: 907 uPSI_JvaScrollText,                   //mX4
21858: 908 uPSI_JvArrow,                         //mX4
21859: 909 unit uPSI_UrlMon;                     //mX4
21860: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'  //mX4
21861: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain;  //DFF
21862: 912 unit uPSI_DFFUtils;                    //DFF
```

```
21863: 913 unit uPSI_MathsLib;                    //DFF
21864: 914 uPSI_UIntList;                        //DFF
21865: 915 uPSI_UGetParens;                      //DFF
21866: 916 unit uPSI_UGeometry;                  //DFF
21867: 917 unit uPSI_UAstronomy;                 //DFF
21868: 918 unit uPSI_UCardComponentV2;           //DFF
21869: 919 unit uPSI_UTGraphSearch;              //DFF
21870: 920 unit uPSI_UParser10;                  //DFF
21871: 921 unit uPSI_cyIEUtils;                  //cY
21872: 922 unit uPSI_UcomboV2;                   //DFF
21873: 923 uPSI_cyBaseComm,                      //cY
21874: 924 uPSI_cyAppInstances,                  //cY
21875: 925 uPSI_cyAttract,                       //cY
21876: 926 uPSI_cyDERUtils                       //cY
21877: 927 unit uPSI_cyDocER;                    //cY
21878: 928 unit uPSI_ODBC;                       //mX
21879: 929 unit uPSI_AssocExec;                  //mX
21880: 930 uPSI_cyBaseCommRoomConnector,         //cY
21881: 931 uPSI_cyCommRoomConnector,             //cY
21882: 932 uPSI_cyCommunicate,                   //cY
21883: 933 uPSI_cyImage;                         //cY
21884: 934 uPSI_cyBaseContainer                  //cY
21885: 935 uPSI_cyModalContainer,                //cY
21886: 936 uPSI_cyFlyingContainer;               //cY
21887: 937 uPSI_RegStr,                          //VCL
21888: 938 uPSI_HtmlHelpViewer;                  //VCL
21889: 939 unit uPSI_cyIniForm                   //cY
21890: 940 unit uPSI_cyVirtualGrid;              //cY
21891: 941 uPSI_Profiler,                        //DA
21892: 942 uPSI_BackgroundWorker,                //DA
21893: 943 uPSI_WavePlay,                        //DA
21894: 944 uPSI_WaveTimer,                       //DA
21895: 945 uPSI_WaveUtils;                       //DA
21896: 946 uPSI_NamedPipes,                      //TB
21897: 947 uPSI_NamedPipeServer,                 //TB
21898: 948 unit uPSI_process,                    //TB
21899: 949 unit uPSI_DPUtils;                    //TB
21900: 950 unit uPSI_CommonTools;                //TB
21901: 951 uPSI_DataSendToWeb,                   //TB
21902: 952 uPSI_StarCalc,                        //TB
21903: 953 uPSI_D2_XPVistaHelperU                //TB
21904: 954 unit uPSI_NetTools                    //TB
21905: 955 unit uPSI_Pipes                       //TB
21906: 956 uPSI_ProcessUnit,                     //mX
21907: 957 uPSI_adGSM,                           //mX
21908: 958 unit uPSI_BetterADODataSet;           //mX
21909: 959 unit uPSI_AdSelCom; //FTT            //mX
21910: 960 unit unit uPSI_dwsXPlatform;          //DWS
21911: 961 uPSI_AdSocket;                        //mX Turbo Power
21912: 962 uPSI_AdPacket;                        //mX
21913: 963 uPSI_AdPort;                          //mX
21914: 964 uPSI_PathFunc;                        //Inno
21915: 965 uPSI_CmnFunc;                        //Inno
21916: 966 uPSI_CmnFunc2; //Inno Setup          //Inno
21917: 967 unit uPSI_BitmapImage;                //mX4
21918: 968 unit uPSI_ImageGrabber;               //mX4
21919: 969 uPSI_SecurityFunc,                    //Inno
21920: 970 uPSI_RedirFunc,                       //Inno
21921: 971 uPSI_FIFO, (MemoryStream)             //mX4
21922: 972 uPSI_Int64Em,                         //Inno
21923: 973 unit uPSI_InstFunc;                   //Inno
21924: 974 unit uPSI_LibFusion;                  //Inno
21925: 975 uPSI_SimpleExpression;                //Inno
21926: 976 uPSI_unitResourceDetails,             //XN
21927: 977 uPSI_unitResFile,                     //XN
21928: 978 unit uPSI_simpleComport;              //mX4
21929: 979 unit uPSI_AfViewershelpers;           //Async
21930: 980 unit uPSI_Console;                    //mX4
21931: 981 unit uPSI_AnalogMeter;                //TB
21932: 982 unit uPSI_XPrinter,                   //TB
21933: 983 unit uPSI_IniFiles;                   //VCL
21934: 984 unit uPSI_lazIniFiles;                //FP
21935: 985 uPSI_testutils;                       //FP
21936: 986 uPSI_ToolsUnit; (DBTests)             //FP
21937: 987 uPSI_fpcunit                          //FP
21938: 988 uPSI_testdecorator;                   //FP
21939: 989 unit uPSI_fpcunittests;               //FP
21940: 990 unit uPSI_cTCPBuffer;                 //Fundamentals 4
21941: 991 unit uPSI_Glut,                       //Open GL
21942: 992 uPSI_LEDBitmaps,                      //mX4
21943: 993 uPSI_FileClass,                       //Inno
21944: 994 uPSI_FileUtilsClass,                  //mX4
21945: 995 uPSI_ComPortInterface; //Kit          //mX4
21946: 996 unit uPSI_SwitchLed;                  //mX4
21947: 997 unit uPSI_cyDmmCanvas;                //cY
21948: 998 uPSI_uColorFunctions;                 //DFF
21949: 999 uPSI_uSettings;                       //DFF
21950: 1000 uPSI_cyDebug.pas                     //cY
21951: 1001 uPSI_cyColorMatrix;                  //cY
```

```
21952: 1002 unit uPSI_cyCopyFiles;                  //cY
21953: 1003 unit uPSI_cySearchFiles;                //cY
21954: 1004 unit uPSI_cyBaseMeasure;                //cY
21955: 1005 unit uPSI_PJIStreams;                   //DD
21956:
21957:
21958: ////////////////////////////////////////////////////////////////////////////
21959: //Form Template Library FTL
21960: ////////////////////////////////////////////////////////////////////////////
21961:
21962: 32 FTL For Form Building out of the Script, eg. 399_form_templates.txt
21963:
21964: 045 unit uPSI_VListView                       TFormListView;
21965: 263 unit uPSI_JvProfiler32;                   TProfReport
21966: 270 unit uPSI_ImgList;                        TCustomImageList
21967: 278 unit uPSI_JvImageWindow;                  TJvImageWindow
21968: 317 unit uPSI_JvParserForm;                   TJvHTMLParserForm
21969: 497 unit uPSI_DebugBox;                       TDebugBox
21970: 513 unit uPSI_ImageWin;                       TImageForm, TImageForm2
21971: 514 unit uPSI_CustomDrawTreeView;             TCustomDrawForm
21972: 515 unit uPSI_GraphWin;                       TGraphWinForm
21973: 516 unit uPSI_actionMain;                     TActionForm
21974: 518 unit uPSI_CtlPanel;                       TAppletApplication
21975: 529 unit uPSI_MDIEdit;                       TEditForm
21976: 535 unit uPSI_CoolMain; {browser}            TWebMainForm
21977: 538 unit uPSI_frmExportMain;                 TSynexportForm
21978: 585 unit uPSI_usniffer; {//PortScanForm}     TSniffForm
21979: 600 unit uPSI_ThreadForm;                    TThreadSortForm;
21980: 618 unit uPSI_delphi_arduino_Unit1;          TLEDForm
21981: 620 unit uPSI_fplotMain;                     TfplotForm1
21982: 660 unit uPSI_JvDBQueryParamsForm;           TJvQueryParamsDialog
21983: 677 unit uPSI_OptionsFrm;                    TfrmOptions;
21984: 718 unit uPSI_MonForm;                       TMonitorForm
21985: 742 unit uPSI_SimplePortMain;               TPortForm1
21986: 770 unit uPSI_ovccalc;                      TOvcCalculator  //widget
21987: 810 unit uPSI_DbExcept;                     TDbEngineErrorDlg
21988: 812 unit uPSI_GL_actorUnit1;                TglActorForm1  //OpenGL Robot
21989: 846 unit uPSI_synhttp_daemon;               TTCPHttpDaemon, TTCPHttpThrd, TPingThread
21990: 867 unit uPSI_Debug;                        TfrmDebug
21991: 904 unit uPSI_U_HexView;                    THexForm2
21992: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain) TOscfrmMain
21993: 959 unit uPSI_AdSelCom;                     TComSelectForm
21994:
21995:
21996: ex.:with TEditForm.create(self) do begin
21997:       caption:= 'Template Form Tester';
21998:       FormStyle:= fsStayOnTop;
21999:       with editor do begin
22000:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
22001:         SelStart:= 0;
22002:         Modified:= False;
22003:       end;
22004:     end;
22005:   with TWebMainForm.create(self) do begin
22006:     URLs.Text:= 'http://www.kleiner.ch';
22007:     URLsClick(self); Show;
22008:   end;
22009:   with TSynexportForm.create(self) do begin
22010:     Caption:= 'Synexport HTML RTF tester';
22011:     Show;
22012:   end;
22013:   with TThreadSortForm.create(self) do begin
22014:      showmodal; free;
22015:   end;
22016:   with TCustomDrawForm.create(self) do begin
22017:        width:=820; height:=820;
22018:        image1.height:= 600; //add properties
22019:        image1.picture.bitmap:= image2.picture.bitmap;
22020:        //SelectionBackground1Click(self) CustomDraw1Click(self);
22021:        Background1.click;
22022:        bitmap1.click;
22023:        Tile1.click;
22024:        Showmodal;
22025:        Free;
22026:      end;
22027:    with TfplotForm1.Create(self) do begin
22028:      BtnPlotClick(self);
22029:      Showmodal; Free;
22030:    end;
22031:   with TOvcCalculator.create(self) do begin
22032:     parent:= aForm;
22033:    //free;
22034:     setbounds(550,510,200,150);
22035:     displaystr:= 'maXcalc';
22036:   end;
22037:   with THexForm2.Create(self) do begin
22038:      ShowModal;
22039:      Free;
22040:   end;
```

```
22041:
22042: function CheckBox: string;
22043: var idHTTP: TIDHTTP;
22044:  begin
22045:    result:= 'version not found';
22046:    if IsInternet then begin
22047:      idHTTP:= TIdHTTP.Create(NIL);
22048:      try
22049:        result:= idHTTP.Get(MXVERSIONFILE2);
22050:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
22051:        if result = MBVER2 then begin
22052:          //output.Font.Style:= [fsbold];
22053:          //Speak(' A new Version '+vstr+' of max box is available! ');
22054:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
22055:        end;
22056:       //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
22057:      finally
22058:        idHTTP.Free
22059:    end;
22060:  end;
22061: end;
22062:
22063:
22064: ////////////////////////////////////////////////////////////////////////////
22065: All maXbox Tutorials Table of Content 2014
22066: ////////////////////////////////////////////////////////////////////////////
22067:     Tutorial 00 Function-Coding  (Blix the Programmer)
22068:     Tutorial 01 Procedural-Coding
22069:     Tutorial 02 OO-Programming
22070:     Tutorial 03 Modular Coding
22071:     Tutorial 04 UML Use Case Coding
22072:     Tutorial 05 Internet Coding
22073:     Tutorial 06 Network Coding
22074:     Tutorial 07 Game Graphics Coding
22075:     Tutorial 08 Operating System Coding
22076:     Tutorial 09 Database Coding
22077:     Tutorial 10 Statistic Coding
22078:     Tutorial 11 Forms Coding
22079:     Tutorial 12 SQL DB Coding
22080:     Tutorial 13 Crypto Coding
22081:     Tutorial 14 Parallel Coding
22082:     Tutorial 15 Serial RS232 Coding
22083:     Tutorial 16 Event Driven Coding
22084:     Tutorial 17 Web Server Coding
22085:     Tutorial 18 Arduino System Coding
22086:     Tutorial 18_3 RGB LED System Coding
22087:     Tutorial 19 WinCOM /Arduino Coding
22088:     Tutorial 20 Regular Expressions RegEx
22089:     Tutorial 21 Android Coding (coming 2013)
22090:     Tutorial 22 Services Programming
22091:     Tutorial 23 Real Time Systems
22092:     Tutorial 24 Clean Code
22093:     Tutorial 25 maXbox Configuration I+II
22094:     Tutorial 26 Socket Programming with TCP
22095:     Tutorial 27 XML & TreeView
22096:     Tutorial 28 DLL Coding (available)
22097:     Tutorial 29 UML Scripting (2014)
22098:     Tutorial 30 Web of Things (2014)
22099:     Tutorial 31 Closures (2014)
22100:     Tutorial 32 SQL Firebird (coming 2014)
22101:     Tutorial 33 Oscilloscope (coming 2015)
22102:     Tutorial 34 GPS Navigation (2014)
22103:     Tutorial 35 Web Box (available)
22104:     Tutorial 36 Unit Testing (coming 2015)
22105:     Tutorial 37 API Coding (coming 2015)
22106:     Tutorial 38 3D Coding (coming 2015)
22107:
22108:
22109: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
22110: using Docu for this file is maxbox_functions_all.pdf
22111: PEP - Pascal Education Program  Lib Lab ShellHell
22112:
22113: http://stackoverflow.com/tags/pascalscript/hot
22114: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
22115: http://sourceforge.net/projects/maXbox   #locs:51620
22116: http://sourceforge.net/apps/mediawiki/maXbox
22117: http://www.blaisepascal.eu/
22118: https://github.com/maxkleiner/maXbox3.git
22119: http://www.heise.de/download/maxbox-1176464.html
22120: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
22121: https://www.facebook.com/pages/Programming-maXbox/166844836691703
22122: http://www.softwareschule.ch/arduino_training.pdf
22123: http://www.delphiarea.com
22124:
22125: All maXbox Examples List
22126: https://github.com/maxkleiner/maXbox3/releases
22127: ********************************************************************************************
22128: 000_pas_baseconvert.txt                          282_fadengraphik.txt
22129: 000_pas_baseconvert.txt_encrypt                  283_SQL_API_messagetimeout.txt
```

22130: 000_pas_baseconvert.txt_decrypt
22131: 001_1_pas_functest - Kopie.txt
22132: 001_1_pas_functest.txt
22133: 001_1_pas_functest2.txt
22134: 001_1_pas_functest_clx2.txt
22135: 001_1_pas_functest_clx2_2.txt
22136: 001_1_pas_functest_openarray.txt
22137: 001_pas_lottogen.txt
22138: 001_pas_lottogen_template.txt
22139: 001_pas_lottogen.txtcopy
22140: 002_pas_russianroulette.txt
22141: 002_pas_russianroulette.txtcopy
22142: 002_pas_russianroulette.txtcopy_decrypt
22143: 002_pas_russianroulette.txtcopy_encrypt
22144: 003_pas_motion.txt
22145: 003_pas_motion.txtcopy
22146: 004_pas_search.txt
22147: 004_pas_search_replace.txt
22148: 004_search_replace_allfunctionlist.txt
22149: 005_pas_oodesign.txt
22150: 005_pas_shelllink.txt
22151: 006_pas_oobatch.txt
22152: 007_pas_streamcopy.txt
22153: 008_EINMALEINS_FUNC.TXT
22154: 008_explanation.txt
22155: 008_pas_verwechselt.txt
22156: 008_pas_verwechselt_ibz_bern_func.txt
22157: 008_stack_ibz.TXT
22158: 009_pas_umlrunner.txt
22159: 009_pas_umlrunner_all.txt
22160: 009_pas_umlrunner_componenttest.txt
22161: 009_pas_umlrunner_solution.txt
22162: 009_pas_umlrunner_solution_2step.txt
22163: 010_pas_oodesign_solution.txt
22164: 011_pas_puzzlepas_defect.txt
22165: 012_pas_umlrunner_solution.txt
22166: 012_pas_umlrunner_solution2.txt
22167: 013_pas_linenumber.txt
22168: 014_pas_primetest.txt
22169: 014_pas_primetest_first.txt
22170: 014_pas_primetest_sync.txt
22171: 015_pas_designbycontract.txt
22172: 015_pas_designbycontract_solution.txt
22173: 016_pas_searchrec.txt
22174: 017_chartgen.txt
22175: 018_data_simulator.txt
22176: 019_dez_to_bin.txt
22177: 019_dez_to_bin_grenzwert_ibz.txt
22178: 020_proc_feedback.txt
22179: 021_pas_symkey.txt
22180: 021_pas_symkey_solution.txt
22181: 022_pas_filestreams.txt
22182: 023_pas_find_searchrec.txt
22183: 023_pas_pathfind.txt
22184: 024_pas_TFileStream_records.txt
22185: 025_prime_direct.txt
22186: 026_pas_memorystream.txt
22187: 027_pas_shellexecute_beta.txt
22188: 027_pas_shellexecute_solution.txt
22189: 028_pas_dataset.txt
22190: 029_pas_assignfile.txt
22191: 029_pas_assignfile_dragndropexe.txt
22192: 030_palindrome_2.txt
22193: 030_palindrome_tester.txt
22194: 030_pas_recursion.txt
22195: 030_pas_recursion2.txt
22196: 031_pas_hashcode.txt
22197: 032_pas_crc_const.txt
22198: 033_pas_cipher.txt
22199: 033_pas_cipher_def.txt
22200: 033_pas_cipher_file_2_solution.txt
22201: 034_pas_soundbox.txt
22202: 035_pas_crcscript.txt
22203: 035_pas_CRCscript_modbus.txt
22204: 036_pas_includetest.txt
22205: 036_pas_includetest_basta.txt
22206: 037_pas_define_demo32.txt
22207: 038_pas_box_demonstrator.txt
22208: 039_pas_dllcall.txt
22209: 040_paspointer.txt
22210: 040_paspointer_old.txt
22211: 041_pasplotter.txt
22212: 041_pasplotter_plus.txt
22213: 042_pas_kgv_ggt.txt
22214: 043_pas_proceduretype.txt
22215: 044_pas_14queens_solwith14.txt
22216: 044_pas_8queens.txt
22217: 044_pas_8queens_sol2.txt
22218: 044_pas_8queens_solutions.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
310_regex_decorator.TXT
312_ListView.txt

22219: 044_queens_performer.txt
22220: 044_queens_performer2.txt
22221: 044_queens_performer2tester.txt
22222: 045_pas_listhandling.txt
22223: 046_pas_records.txt
22224: 047_pas_modula10.txt
22225: 048_pas_romans.txt
22226: 049_pas_ifdemo.txt
22227: 049_pas_ifdemo_BROKER.txt
22228: 050_pas_primetest2.txt
22229: 050_pas_primetester_thieves.txt
22230: 050_program_starter.txt
22231: 050_program_starter_performance.txt
22232: 051_pas_findtext_solution.txt
22233: 052_pas_text_as_stream.txt
22234: 052_pas_text_as_stream_include.txt
22235: 053_pas_singleton.txt
22236: 054_pas_speakpassword.txt
22237: 054_pas_speakpassword2.txt
22238: 054_pas_speakpassword_searchtest.txt
22239: 055_pas_factorylist.txt
22240: 056_pas_demeter.txt
22241: 057_pas_dirfinder.txt
22242: 058_pas_filefinder.txt
22243: 058_pas_filefinder_pdf.txt
22244: 058_pas_filefinder_screview.txt
22245: 058_pas_filefinder_screview2.txt
22246: 058_pas_filefinder_screview3.txt
22247: 059_pas_timertest.txt
22248: 059_pas_timertest_2.txt
22249: 059_pas_timertest_time_solution.txt
22250: 059_timerobject_starter2.txt
22251: 059_timerobject_starter2_ibz2_async.txt
22252: 059_timerobject_starter2_uml.txt
22253: 059_timerobject_starter2_uml_main.txt
22254: 059_timerobject_starter4_ibz.txt
22255: 060_pas_datefind.txt
22256: 060_pas_datefind_exceptions2.txt
22257: 060_pas_datefind_exceptions_CHECKTEST.txt
22258: 060_pas_datefind_fulltext.txt
22259: 060_pas_datefind_plus.txt
22260: 060_pas_datefind_plus_mydate.txt
22261: 061_pas_randomwalk.txt
22262: 061_pas_randomwalk_plus.txt
22263: 062_paskorrelation.txt
22264: 063_pas_calculateform.txt
22265: 063_pas_calculateform_2list.txt
22266: 064_pas_timetest.txt
22267: 065_pas_bitcounter.txt
22268: 066_pas_eliza.txt
22269: 066_pas_eliza_include_sol.txt
22270: 067_pas_morse.txt
22271: 068_pas_piezo_sound.txt
22272: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
22273: 069_my_LEDBOX.TXT
22274: 069_pas_ledmatrix.txt
22275: 069_pas_LEDMATRIX_Alphabet.txt
22276: 069_pas_LEDMATRIX_Alphabet_run.txt
22277: 069_pas_LEDMATRIX_Alphabet_tester.txt
22278: 069_PAS_LEDMATRIX_COLOR.TXT
22279: 069_pas_ledmatrix_fixedit.txt
22280: 069_pas_LEDMATRIX_soundbox.txt
22281: 069_pas_LEDMATRIX_soundbox2.txt
22282: 069_Richter_MATRIX.TXT
22283: 070_pas_functionplot.txt
22284: 070_pas_functionplotter2.txt
22285: 070_pas_functionplotter2_mx4.txt
22286: 070_pas_functionplotter2_tester.txt
22287: 070_pas_functionplotter3.txt
22288: 070_pas_functionplotter4.txt
22289: 070_pas_functionplotter_digital.txt
22290: 070_pas_functionplotter_elliptic.txt
22291: 070_pas_function_helmholtz.txt
22292: 070_pas_textcheck_experimental.txt
22293: 071_pas_graphics.txt
22294: 071_pas_graphics_drawsym.txt
22295: 071_pas_graphics_drawsym_save.txt
22296: 071_pas_graphics_random.txt
22297: 072_pas_fractals.txt
22298: 072_pas_fractals_2.txt
22299: 072_pas_fractals_blackhole.txt
22300: 072_pas_fractals_perfomance.txt
22301: 072_pas_fractals_perfomance_new.txt
22302: 072_pas_fractals_perfomance_sharp.txt
22303: 072_pas_fractals_performance.txt
22304: 072_pas_fractals_performance_mX4.txt
22305: 073_pas_forms.txt
22306: 074_pas_chartgenerator.txt
22307: 074_pas_chartgenerator_solution.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
362_maxonmotor2.TXT
363_compress_services.txt

22308: 074_pas_chartgenerator_solution_back.txt
22309: 074_pas_charts.txt
22310: 075_bitmap_Artwork2.txt
22311: 075_pas_bitmappuzzle.txt
22312: 075_pas_bitmappuzzle24.prod.txt
22313: 075_pas_bitmappuzzle2_prod.txt
22314: 075_pas_bitmappuzzle3.txt
22315: 075_pas_bitmapsolve.txt
22316: 075_pas_bitmap_Artwork.txt
22317: 075_pas_puzzlepas_solution.txt
22318: 076_pas_3dcube.txt
22319: 076_pas_circle.txt
22320: 077_pas_mmshow.txt
22321: 078_pas_pi.txt
22322: 079_pas_3dcube_animation.txt
22323: 079_pas_3dcube_animation4.txt
22324: 079_pas_3dcube_plus.txt
22325: 080_pas_hanoi.txt
22326: 080_pas_hanoi2.txt
22327: 080_pas_hanoi2_file.txt
22328: 080_pas_hanoi2_sol.txt
22329: 080_pas_hanoi2_tester.txt
22330: 080_pas_hanoi2_tester_fast.txt
22331: 080_pas_hanoi3.txt
22332: 081_pas_chartist2.txt
22333: 082_pas_biorythmus.txt
22334: 082_pas_biorythmus_solution.txt
22335: 082_pas_biorythmus_solution_3.txt
22336: 082_pas_biorythmus_test.txt
22337: 083_pas_GITARRE.txt
22338: 083_pas_soundbox_tones.txt
22339: 084_pas_waves.txt
22340: 085_mxsinus_logo.txt
22341: 085_sinus_plot_waves.txt
22342: 086_pas_graph_arrow_heart.txt
22343: 087_bitmap_loader.txt
22344: 087_pas_bitmap_solution.txt
22345: 087_pas_bitmap_solution2.txt
22346: 087_pas_bitmap_subimage.txt
22347: 087_pas_bitmap_test.txt
22348: 088_pas_soundbox2_mp3.txt
22349: 088_pas_soundbox_mp3.txt
22350: 088_pas_sphere_2.txt
22351: 089_pas_gradient.txt
22352: 089_pas_maxland2.txt
22353: 090_pas_sudoku4.txt
22354: 090_pas_sudoku4_2.txt
22355: 091_pas_cube4.txt
22356: 092_pas_statistics4.txt
22357: 093_variance.txt
22358: 093_variance_debug.txt
22359: 094_pas_daysold.txt
22360: 094_pas_stat_date.txt
22361: 095_pas_ki_simulation.txt
22362: 096_pas_geisen_problem.txt
22363: 096_pas_montyhall_problem.txt
22364: 097_lotto_proofofconcept.txt
22365: 097_pas_lottocombinations_beat_plus.txt
22366: 097_pas_lottocombinations_beat_plus2.txt
22367: 097_pas_lottocombinations_universal.txt
22368: 097_pas_lottosimulation.txt
22369: 098_pas_chartgenerator_plus.txt
22370: 099_pas_3D_show.txt
22371: 200_big_numbers.txt
22372: 200_big_numbers2.txt
22373: 201_streamload_xml.txt
22374: 202_systemcheck.txt
22375: 203_webservice_simple_intftester.txt
22376: 204_webservice_simple.txt
22377: 205_future_value_service.txt
22378: 206_DTD_string_functions.txt
22379: 207_ibz2_async_process.txt
22380: 208_crc32_hash.txt
22381: 209_cryptohash.txt
22382: 210_public_private.txt
22383: 210_public_private_cryptosystem.txt
22384: 211_wipe_pattern.txt
22385: 211_wipe_pattern2.txt
22386: 211_wipe_pattern_solution.txt
22387: 212_pas_statisticmodule4.TXT
22388: 212_pas_statisticmoduletxt.TXT
22389: 212_statisticmodule4.txt
22390: 213_pas_BBP_Algo.txt
22391: 214_mxdocudemo.txt
22392: 214_mxdocudemo2.txt
22393: 214_mxdocudemo3.txt
22394: 215_hints_test.TXT
22395: 216_warnings_test.TXT
22396: 217_pas_heartbeat.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
400_fplottchart2.TXT
400_fplottchart2teetest.TXT

22397: 218_biorythmus_studio.txt
22398: 219_cipherbox.txt
22399: 219_crypt_source_comtest_solution.TXT
22400: 220_cipherbox_form.txt
22401: 220_cipherbox_form2.txt
22402: 221_bcd_explain.txt
22403: 222_memoform.txt
22404: 223_directorybox.txt
22405: 224_dialogs.txt
22406: 225_sprite_animation.txt
22407: 226_ASCII_Grid2.TXT
22408: 227_animation.txt
22409: 227_animation2.txt
22410: 228_android_calendar.txt
22411: 229_android_game.txt
22412: 229_android_game_tester.txt
22413: 230_DataProvider.txt
22414: 230_DataSetProvider.txt
22415: 230_DataSetXMLBackupScholz.txt
22416: 231_DBGrid_access.txt
22417: 231_DBGrid_XMLaccess.txt
22418: 231_DBGrid_XMLaccess2.txt
22419: 231_DBGrid_XMLaccess_locatetester.txt
22420: 231_DBGrid_XML_CDS_local.txt
22421: 232_outline.txt
22422: 232_outline_2.txt
22423: 233_modular_form.txt
22424: 234_debugoutform.txt
22425: 235_fastform.TXT
22426: 236_componentpower.txt
22427: 236_componentpower_back.txt
22428: 237_pas_4forms.txt
22429: 238_lottogen_form.txt
22430: 239_pas_sierpinski.txt
22431: 239_pas_sierpinski2.txt
22432: 240_unitGlobal_tester.txt
22433: 241_db3_sql_tutorial.txt
22434: 241_db3_sql_tutorial2.txt
22435: 241_db3_sql_tutorial2fix.txt
22436: 241_db3_sql_tutorial3.txt
22437: 241_db3_sql_tutorial3connect.txt
22438: 241_db3_sql_tutorial3_fptest.txt
22439: 241_RTL_SET2.txt
22440: 241_RTL_SET2_tester.txt
22441: 242_Component_Control.txt
22442: 243_tutorial_loader.txt
22443: 244_script_loader_loop.txt
22444: 245_formapp2.txt
22445: 245_formapp2_tester.txt
22446: 245_formapp2_testerX.txt
22447: 246_httpapp.txt
22448: 247_datecalendar.txt
22449: 248_ASCII_Grid2_sorted.TXT
22450: 249_picture_grid.TXT
22451: 250_tipsandtricks2.txt
22452: 250_tipsandtricks3.txt
22453: 250_tipsandtricks3api.txt
22454: 250_tipsandtricks3_admin_elevation.txt
22455: 250_tipsandtricks3_tester.txt
22456: 250_tipsandtricks4_tester.txt
22457: 250_tipsandtricks4_tester2.txt
22458: 251_compare_noise_gauss.txt
22459: 251_whitenoise.txt
22460: 251_whitenoise2.txt
22461: 252_hilbert_turtle.txt
22462: 252_pas_hilbert.txt
22463: 253_opearatingsystem3.txt
22464: 254_dynarrays.txt
22465: 255_einstein.txt
22466: 256_findconsts_of_EXE.txt
22467: 256_findfunctions2_of_EXE.txt
22468: 256_findfunctions2_of_EXEaverp.txt
22469: 256_findfunctions2_of_EXEspec.txt
22470: 256_findfunctions3.txt
22471: 256_findfunctions_of_EXE.txt
22472: 257_AES_Cipher.txt
22473: 258_AES_cryptobox.txt
22474: 258_AES_cryptobox2.txt
22475: 258_AES_cryptobox2_passdlg.txt
22476: 259_AES_crypt_directory.txt
22477: 260_sendmessage_2.TXT
22478: 260_sendmessage_beta.TXT
22479: 261_probability.txt
22480: 262_mxoutputdemo4.txt
22481: 263_async_sound.txt
22482: 264_vclutils.txt
22483: 264_VCL_utils2.txt
22484: 265_timer_API.txt
22485: 266_serial_interface.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
458_atomimageX.txt
459_cindyfunc.txt

```
22486: 266_serial_interface2.txt                    459_cindyfunc2.txt
22487: 266_serial_interface3.txt                    460_TopTenFunctions.txt
22488: 267_ackermann_rec.txt                        461_sqlform_calwin.txt
22489: 267_ackermann_variants.txt                   462_caesarcipher.txt
22490: 268_DBGrid_tree.txt                          463_global_exception.txt
22491: 269_record_grid.TXT                          464_function_procedure.txt
22492: 270_Jedi_FunctionPower.txt                   464_function_procedure2.txt
22493: 270_Jedi_FunctionPowertester.txt             464_function_procedure3.txt
22494: 271_closures_study.txt                       465_U_HexView.txt
22495: 271_closures_study_workingset2.txt           466_moon.txt
22496: 272_pas_function_show.txt                    466_moon_inputquery.txt
22497: 273_pas_function_show2.txt                   4671_cardmagic.txt
22498: 274_library_functions.txt                    467_helmholtz_graphic.txt
22499: 275_turtle_language.txt                      468_URLMon.txt
22500: 275_turtle_language_save.txt                 468_URLMon2.txt
22501: 276_save_algo.txt                           469_formarrow.txt
22502: 276_save_algo2.txt                          469_formarrow_datepicker.txt
22503: 277_functionsfor39.txt                      469_formarrow_datepicker_ibz_result.txt
22504: 278_DB_Dialogs.TXT                          469_ibzresult.txt
22505: 279_hexer2.TXT                              470_DFFUtils_compiled.txt
22506: 279_hexer2macro.TXT                         470_DFFUtils_ScrollingLED.txt
22507: 279_hexer2macroback.TXT                     470_Oscilloscope.txt
22508: 280_UML_process.txt                         470_Oscilloscope_code.txt
22509: 280_UML_process_knabe2.txt                  471_cardmagic.txt
22510: 280_UML_process_knabe3.txt                  471_cardmagic2.txt
22511: 280_UML_process_TIM_Botzenhardt.txt         472_allcards.TXT
22512: 280_UML_TIM_Seitz.txt                       473_comboset.txt
22513: 281_picturepuzzle.txt                       474_wakeonlan.txt
22514: 281_picturepuzzle2.txt                      474_wakeonlan2.txt
22515: 281_picturepuzzle3.txt                      476_getscripttest.txt
22516: 281_picturepuzzle4.txt                      477_filenameonly.txt
22517: 479_inputquery.txt                          480_regex_pathfinder.txt
22518: 480_regex_pathfinder2.txt                   481_processList.txt
22519: 482 processPipe.txt                         482 processPipeGCC.txt
22520: 483_PathFuncTest_mX.txt                     484_filefinder3.txt
22521: 485_InnoFunc.txt                            486_VideoGrabber.txt
22522: 487_asyncKeyState.txt                       488_asyncTerminal.txt
22523: 489_simpleComport.txt                       490_webCamproc.txt
22524: 491_analogmeter.txt                         492_snowflake2.txt
22525: 493_gadgets.txt                             495_fourierfreq.txt
22526: 496_InstallX.txt                            497_LED.txt
22527: 498_UnitTesting.txt                         499_mulu42.txt
22528: 500_diceoflifes.txt                         501_firebird_datasnap_tests.txt
22529: 502_findalldocs.txt                         503_led_switch.txt
22530: 504_fileclass.txt                           505_debug.txt
22531:
22532:
22533: Web Script Examples:
22534:
22535: http://www.softwareschule.ch/examples/performer.txt';
22536: http://www.softwareschule.ch/examples/turtle.txt';
22537: http://www.softwareschule.ch/examples/SQLExport.txt';
22538: http://www.softwareschule.ch/examples/Richter.txt';
22539: http://www.softwareschule.ch/examples/checker.txt';
22540: http://www.softwareschule.ch/examples/demoscript.txt';
22541: http://www.softwareschule.ch/examples/ibzresult.txt';
22542: http://www.softwareschule.ch/examples/performindex.txt
22543: http://www.softwareschule.ch/examples/processlist.txt
22544: http://www.softwareschule.ch/examples/game.txt
22545:
22546:
22547:
22548: Delphi Basics Run Time Library listing
22549: ********************************************************************************
22550: A
22551: Compiler Directive $A Determines whether data is aligned or packed
22552: Compiler Directive $Align Determines whether data is aligned or packed
22553: Compiler Directive $AppType Determines the application type : GUI or Console
22554: Procedure SysUtils Abort Aborts the current processing with a silent exception
22555: Function System Abs Gives the absolute value of a number (-ve sign is removed)
22556: Directive Abstract Defines a class method only implemented in subclasses
22557: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
22558: Function System Addr Gives the address of a variable, function or procedure
22559: Keyword And Boolean and or bitwise and of two arguments
22560: Type System AnsiChar A character type guaranteed to be 8 bits in size
22561: Function SysUtils AnsiCompareStr Compare two strings for equality
22562: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
22563: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
22564: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
22565: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
22566: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
22567: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
22568: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
22569: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string
22570: Function StrUtils AnsiPos Find the position of one string in another
22571: Function StrUtils AnsiReplaceStr Replaces a part of one string with another
22572: Function StrUtils AnsiReverseString Reverses the sequence of letters in a string
22573: Function StrUtils AnsiRightStr Extracts characters from the right of a string
22574: Function StrUtils AnsiStartsStr Returns true if a string starts with a substring
```

```
22575: Type System AnsiString A data type that holds a string of AnsiChars
22576: Function SysUtils AnsiUpperCase Change lower case characters in a string to upper case
22577: Procedure System Append Open a text file to allow appending of text to the end
22578: Procedure SysUtils AppendStr Concatenate one string onto the end of another
22579: Function Math ArcCos The Arc Cosine of a number, returned in radians
22580: Function Math ArcSin The Arc Sine of a number, returned in radians
22581: Function System ArcTan The Arc Tangent of a number, returned in radians
22582: Keyword Array A data type holding indexable collections of data
22583: Keyword As Used for casting object references
22584: Procedure System Assign Assigns a file handle to a binary or text file
22585: Function System Assigned Returns true if a reference is not nil
22586: Procedure System AssignFile Assigns a file handle to a binary or text file
22587: Procedure Printers AssignPrn Treats the printer as a text file - an easy way of printing text
22588:
22589: B
22590: Compiler Directive $B Whether to short cut and and or operations
22591: Compiler Directive $BoolEval Whether to short cut and and or operations
22592: Procedure SysUtils Beep Make a beep sound
22593: Keyword Begin Keyword that starts a statement block
22594: Function System BeginThread Begins a separate thread of code execution
22595: Procedure System BlockRead Reads a block of data records from an untyped binary file
22596: Procedure System BlockWrite Writes a block of data records to an untyped binary file
22597: Type System Boolean Allows just True and False values
22598: Function Classes Bounds Create a TRect value from top left and size values
22599: Procedure System Break Forces a jump out of a single loop
22600: Type System Byte An integer type supporting values 0 to 255
22601:
22602: C
22603: Type System Cardinal The basic unsigned integer type
22604: Keyword Case A mechanism for acting upon different values of an Ordinal
22605: Function StdConvs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
22606: Function SysUtils ChangeFileExt Change the extension part of a file name
22607: Type System Char Variable type holding a single character
22608: Procedure System ChDir Change the working drive plus path for a specified drive
22609: Function System Chr Convert an integer into a character
22610: Keyword Class Starts the declaration of a type of object class
22611: Procedure System Close Closes an open file
22612: Procedure System CloseFile Closes an open file
22613: Variable System CmdLine Holds the execution text used to start the current program
22614: Type System Comp A 64 bit signed integer
22615: Function SysUtils CompareStr Compare two strings to see which is greater than the other
22616: Function SysUtils CompareText Compare two strings for equality, ignoring case
22617: Function Math CompareValue Compare numeric values with a tolerance
22618: Function System Concat Concatenates one or more strings into one string
22619: Keyword Const Starts the definition of fixed data values
22620: Keyword Constructor Defines the method used to create an object from a class
22621: Procedure System Continue Forces a jump to the next iteration of a loop
22622: Function ConvUtils Convert Convert one measurement value to another
22623: Function System Copy Create a copy of part of a string or an array
22624: Function System Cos The Cosine of a number
22625: Function SysUtils CreateDir Create a directory
22626: Type System Currency A floating point type with 4 decimals used for financial values
22627: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
22628: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
22629: Variable SysUtils CurrencyString The currency string used in currency display functions
22630: Function SysUtils CurrToStr Convert a currency value to a string
22631: Function SysUtils CurrToStrF Convert a currency value to a string with formatting
22632:
22633: D
22634: Compiler Directive $D Determines whether application debug information is built
22635: Compiler Directive $DebugInfo Determines whether application debug information is built
22636: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
22637: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
22638: Function SysUtils Date Gives the current date
22639: Variable SysUtils DateSeparator The character used to separate display date fields
22640: Function SysUtils DateTimeToFileDate Convert a TDateTime value to a File date/time format
22641: Function SysUtils DateTimeToStr Converts TDateTime date and time values to a string
22642: Procedure SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
22643: Function SysUtils DateToStr Converts a TDateTime date value to a string
22644: Function DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
22645: Function DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
22646: Function DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
22647: Function SysUtils DayOfWeek Gives day of week index for a TDateTime value
22648: Function DateUtils DaysBetween Gives the whole number of days between 2 dates
22649: Function DateUtils DaysInAMonth Gives the number of days in a month
22650: Function DateUtils DaysInAYear Gives the number of days in a year
22651: Function DateUtils DaySpan Gives the fractional number of days between 2 dates
22652: Procedure System Dec Decrement an ordinal variable
22653: Variable SysUtils DecimalSeparator The character used to display the decimal point
22654: Procedure SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
22655: Procedure DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
22656: Procedure SysUtils DecodeTime Break a TDateTime value into individual time values
22657: Directive Default Defines default processing for a property
22658: Function Math DegToRad Convert a degrees value to radians
22659: Procedure System Delete Delete a section of characters from a string
22660: Function SysUtils DeleteFile Delete a file specified by its file name
22661: Keyword Destructor Defines the method used to destroy an object
22662: Function SysUtils DirectoryExists Returns true if the given directory exists
22663: Function SysUtils DiskFree Gives the number of free bytes on a specified drive
```

```
22664: Function SysUtils DiskSize Gives the size in bytes of a specified drive
22665: Procedure System Dispose Dispose of storage used by a pointer type variable
22666: Keyword Div Performs integer division, discarding the remainder
22667: Keyword Do Defines the start of some controlled action
22668: Type System Double A floating point type supporting about 15 digits of precision
22669: Keyword DownTo Prefixes an decremental for loop target value
22670: Function StrUtils DupeString Creates a string containing copies of a substring
22671: Directive Dynamic Allows a class method to be overriden in derived classes
22672:
22673: E
22674: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
22675: Compiler Directive $EndIf Terminates conditional code compilation
22676: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
22677: Keyword Else Starts false section of if, case and try statements
22678: Function SysUtils EncodeDate Build a TDateTime value from year, month and day values
22679: Function DateUtils EncodeDateTime Build a TDateTime value from day and time values
22680: Function SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
22681: Keyword End Keyword that terminates statement blocks
22682: Function DateUtils EndOfADay Generate a TDateTime value set to the very end of a day
22683: Function DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
22684: Procedure System EndThread Terminates a thread with an exit code
22685: Function System Eof Returns true if a file opened with Reset is at the end
22686: Function System Eoln Returns true if the current text file is pointing at a line end
22687: Procedure System Erase Erase a file
22688: Variable System ErrorAddr Sets the error address when an application terminates
22689: Keyword Except Starts the error trapping clause of a Try statement
22690: Procedure System Exclude Exclude a value in a set variable
22691: Procedure System Exit Exit abruptly from a function or procedure
22692: Variable System ExitCode Sets the return code when an application terminates
22693: Function System Exp Gives the exponent of a number
22694: Directive System Export Makes a function or procedure in a DLL externally available
22695: Type System Extended The floating point type with the highest capacity and precision
22696: Function SysUtils ExtractFileDir Extracts the dir part of a full file name
22697: Function SysUtils ExtractFileDrive Extracts the drive part of a full file name
22698: Function SysUtils ExtractFileExt Extracts the extension part of a full file name
22699: Function SysUtils ExtractFileName Extracts the name part of a full file name
22700: Function SysUtils ExtractFilePath Extracts the path part of a full file name
22701:
22702: F
22703: Function StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
22704: Keyword File Defines a typed or untyped file
22705: Function SysUtils FileAge Get the last modified date/time of a file without opening it
22706: Function SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
22707: Function SysUtils FileExists Returns true if the given file exists
22708: Function SysUtils FileGetAttr Gets the attributes of a file
22709: Variable System FileMode Defines how Reset opens a binary file
22710: Function System FilePos Gives the file position in a binary or text file
22711: Function SysUtils FileSearch Search for a file in one or more directories
22712: Function SysUtils FileSetAttr Sets the attributes of a file
22713: Function SysUtils FileSetDate Set the last modified date and time of a file
22714: Function System FileSize Gives the size in records of an open file
22715: Procedure System FillChar Fills out a section of storage with a fill character or byte value
22716: Keyword Finally Starts the unconditional code section of a Try statement
22717: Function SysUtils FindClose Closes a successful FindFirst file search
22718: Function SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
22719: Function SysUtils FindFirst Finds all files matching a file mask and attributes
22720: Function SysUtils FindNext Find the next file after a successful FindFirst
22721: Function SysUtils FloatToStr Convert a floating point value to a string
22722: Function SysUtils FloatToStrF Convert a floating point value to a string with formatting
22723: Procedure System Flush Flushes buffered text file data to the file
22724: Keyword For Starts a loop that executes a finite number of times
22725: Function SysUtils ForceDirectories Create a new path of directories
22726: Function SysUtils Format Rich formatting of numbers and text into a string
22727: Function SysUtils FormatCurr Rich formatting of a currency value into a string
22728: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
22729: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
22730: Function System Frac The fractional part of a floating point number
22731: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil
22732: Procedure System FreeMem Free memory storage used by a variable
22733: Keyword System Function Defines a subroutine that returns a value
22734:
22735: G
22736: Function SysUtils GetCurrentDir Get the current directory (drive plus directory)
22737: Procedure System GetDir Get the default directory (drive plus path) for a specified drive
22738: Function System GetLastError Gives the error code of the last failing Windows API call
22739: Procedure SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
22740: Function System GetMem Get a specified number of storage bytes
22741: Keyword Goto Forces a jump to a label, regardless of nesting
22742:
22743: H
22744: Compiler Directive $H Treat string types as AnsiString or ShortString
22745: Compiler Directive $Hints Determines whether Delphi shows compilation hints
22746: Procedure System Halt Terminates the program with an optional dialog
22747: Function System Hi Returns the hi-order byte of a (2 byte) Integer
22748: Function System High Returns the highest value of a type or variable
22749:
22750: I
22751: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
22752: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
```

```
22753: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
22754: Compiler Directive $IfOpt Tests for the state of a Compiler directive
22755: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
22756: Compiler Directive $IOChecks When on, an IO operation error throws an exception
22757: Keyword If Starts a conditional expression to determine what to do next
22758: Keyword Implementation Starts the implementation (code) section of a Unit
22759: Keyword In Used to test if a value is a member of a set
22760: Procedure System Inc Increment an ordinal variable
22761: Function DateUtils IncDay Increments a TDateTime variable by + or - number of days
22762: Procedure System Include Include a value in a set variable
22763: Function DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
22764: Function DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
22765: Function SysUtils IncMonth Increments a TDateTime variable by a number of months
22766: Function DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
22767: Function DateUtils IncYear Increments a TDateTime variable by a number of years
22768: Directive Index Principally defines indexed class data properties
22769: Constant Math Infinity Floating point value of infinite size
22770: Keyword Inherited Used to call the parent class constructor or destructor method
22771: Variable System Input Defines the standard input text file
22772: Function Dialogs InputBox Display a dialog that asks for user text input, with default
22773: Function Dialogs InputQuery Display a dialog that asks for user text input
22774: Procedure System Insert Insert a string into another string
22775: Function System Int The integer part of a floating point number as a float
22776: Type System Int64 A 64 bit sized integer - the largest in Delphi
22777: Type System Integer The basic Integer type
22778: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
22779: Function SysUtils IntToHex Convert an Integer into a hexadecimal string
22780: Function SysUtils IntToStr Convert an integer into a string
22781: Function System IOResult Holds the return code of the last I/O operation
22782: Keyword Is Tests whether an object is a certain class or ascendant
22783: Function Math IsInfinite Checks whether a floating point number is infinite
22784: Function SysUtils IsLeapYear Returns true if a given calendar year is a leap year
22785: Function System IsMultiThread Returns true if the code is running multiple threads
22786: Function Math IsNaN Checks to see if a floating point number holds a real number
22787:
22788: L
22789: Compiler Directive $L Determines what application debug information is built
22790: Compiler Directive $LocalSymbols Determines what application debug information is built
22791: Compiler Directive $LongStrings Treat string types as AnsiString or ShortString
22792: Function SysUtils LastDelimiter Find the last position of selected characters in a string
22793: Function System Length Return the number of elements in an array or string
22794: Function System Ln Gives the natural logarithm of a number
22795: Function System Lo Returns the low-order byte of a (2 byte) Integer
22796: Function Math Log10 Gives the log to base 10 of a number
22797: Variable SysUtils LongDateFormat Long version of the date to string format
22798: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
22799: Type System LongInt An Integer whose size is guaranteed to be 32 bits
22800: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
22801: Variable SysUtils LongTimeFormat Long version of the time to string format
22802: Type System LongWord A 32 bit unsigned integer
22803: Function System Low Returns the lowest value of a type or variable
22804: Function SysUtils LowerCase Change upper case characters in a string to lower case
22805:
22806: M
22807: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
22808: Function Math Max Gives the maximum of two integer values
22809: Constant System MaxInt The maximum value an Integer can have
22810: Constant System MaxLongInt The maximum value an LongInt can have
22811: Function Math Mean Gives the average for a set of numbers
22812: Function Dialogs MessageDlg Displays a message, symbol, and selectable buttons
22813: Function Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
22814: Function Math Min Gives the minimum of two integer values
22815: Constant SysUtils MinsPerDay Gives the number of minutes in a day
22816: Procedure System MkDir Make a directory
22817: Keyword Mod Performs integer division, returning the remainder
22818: Constant SysUtils MonthDays Gives the number of days in a month
22819: Function DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
22820: Procedure System Move Copy bytes of data from a source to a destination
22821:
22822: N
22823: Constant Math NaN Not a real number
22824: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
22825: Procedure System New Create a new pointer type variable
22826: Constant System Nil A pointer value that is defined as undetermined
22827: Keyword Not Boolean Not or bitwise not of one arguments
22828: Function SysUtils Now Gives the current date and time
22829: Variable Variants Null A variable that has no value
22830:
22831: O
22832: Compiler Directive $O Determines whether Delphi optimises code when compiling
22833: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
22834: Compiler Directive $OverFlowChecks Determines whether Delphi checks integer and enum bounds
22835: Keyword System Object Allows a subroutine data type to refer to an object method
22836: Function System Odd Tests whether an integer has an odd value
22837: Keyword Of Linking keyword used in many places
22838: Keyword On Defines exception handling in a Try Except clause
22839: Keyword Or Boolean or or bitwise or of two arguments
22840: Function System Ord Provides the Ordinal value of an integer, character or enum
22841: Directive Out Identifies a routine parameter for output only
```

```
22842: Variable System Output Defines the standard output text file
22843: Directive Overload Allows 2 or more routines to have the same name
22844: Directive Override Defines a method that replaces a virtual parent class method
22845:
22846: P
22847: Keyword Packed Compacts complex data types into minimal storage
22848: Type System PAnsiChar A pointer to an AnsiChar value
22849: Type System PAnsiString Pointer to an AnsiString value
22850: Function System ParamCount Gives the number of parameters passed to the current program
22851: Function System ParamStr Returns one of the parameters used to run the current program
22852: Type System PChar A pointer to an Char value
22853: Type System PCurrency Pointer to a Currency value
22854: Type System PDateTime Pointer to a TDateTime value
22855: Type System PExtended Pointer to a Extended floating point value
22856: Function System Pi The mathematical constant
22857: Type System PInt64 Pointer to an Int64 value
22858: Function Classes Point Generates a TPoint value from X and Y values
22859: Type System Pointer Defines a general use Pointer to any memory based data
22860: Function Classes PointsEqual Compares two TPoint values for equality
22861: Function System Pos Find the position of one string in another
22862: Function System Pred Decrement an ordinal variable
22863: Function Printers Printer Returns a reference to the global Printer object
22864: Directive Private Starts the section of private data and methods in a class
22865: Keyword System Procedure Defines a subroutine that does not return a value
22866: Procedure FileCtrl ProcessPath Split a drive/path/filename string into its constituent parts
22867: Keyword System Program Defines the start of an application
22868: Function Dialogs PromptForFileName Shows a dialog allowing the user to select a file
22869: Keyword System Property Defines controlled access to class fields
22870: Directive Protected Starts a section of class private data accesible to sub-classes
22871: Type System PShortString A pointer to an ShortString value
22872: Type System PString Pointer to a String value
22873: Function Types PtInRect Tests to see if a point lies within a rectangle
22874: Directive Public Starts an externally accessible section of a class
22875: Directive Published Starts a published externally accessible section of a class
22876: Type System PVariant Pointer to a Variant value
22877: Type System PWideChar Pointer to a WideChar
22878: Type System PWideString Pointer to a WideString value
22879:
22880: Q
22881: Compiler Directive $Q Determines whether Delphi checks integer and enum bounds
22882:
22883: R
22884: Compiler Directive $R Determines whether Delphi checks array bounds
22885: Compiler Directive $RangeChecks Determines whether Delphi checks array bounds
22886: Compiler Directive $ReferenceInfo Determines whether symbol reference information is built
22887: Compiler Directive $Resource Defines a resource file to be included in the application linking
22888: Function Math RadToDeg Converts a radian value to degrees
22889: Keyword Raise Raise an exception
22890: Function System Random Generate a random floating point or integer number
22891: Procedure System Randomize Reposition the Random number generator next value
22892: Function Math RandomRange Generate a random integer number within a supplied range
22893: Variable System RandSeed Reposition the Random number generator next value
22894: Procedure System Read Read data from a binary or text file
22895: Procedure System ReadLn Read a complete line of data from a text file
22896: Type System Real A floating point type supporting about 15 digits of precision
22897: Type System Real48 The floating point type with the highest capacity and precision
22898: Procedure System ReallocMem Reallocate an existing block of storage
22899: Function DateUtils RecodeDate Change only the date part of a TDateTime variable
22900: Function DateUtils RecodeTime Change only the time part of a TDateTime variable
22901: Keyword Record A structured data type - holding fields of data
22902: Function Classes Rect Create a TRect value from 2 points or 4 coordinates
22903: Function SysUtils RemoveDir Remove a directory
22904: Procedure System Rename Rename a file
22905: Function SysUtils RenameFile Rename a file or directory
22906: Keyword Repeat Repeat statements until a ternmination condition is met
22907: Procedure SysUtils ReplaceDate Change only the date part of a TDateTime variable
22908: Procedure SysUtils ReplaceTime Change only the time part of a TDateTime variable
22909: Procedure System Reset Open a text file for reading, or binary file for read/write
22910: Variable System Result A variable used to hold the return value from a function
22911: Procedure System ReWrite Open a text or binary file for write access
22912: Procedure System RmDir Remove a directory
22913: Function System Round Rounds a floating point number to an integer
22914: Procedure System RunError Terminates the program with an error dialog
22915:
22916: S
22917: Constant SysUtils SecsPerDay Gives the number of seconds in a day
22918: Procedure System Seek Move the pointer in a binary file to a new record position
22919: Function System SeekEof Skip to the end of the current line or file
22920: Function System SeekEoln Skip to the end of the current line or file
22921: Function FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
22922: Variable System Self Hidden parameter to a method - refers to the containing object
22923: Keyword Set Defines a set of up to 255 distinct values
22924: Function SysUtils SetCurrentDir Change the current directory
22925: Procedure System SetLength Changes the size of a string, or the size(s) of an array
22926: Procedure System SetString Copies characters from a buffer into a string
22927: Keyword Shl Shift an integer value left by a number of bits
22928: Variable SysUtils ShortDateFormat Compact version of the date to string format
22929: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
22930: Type System ShortInt An integer type supporting values -128 to 127
```

```
22931: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
22932: Type System ShortString Defines a string of up to 255 characters
22933: Variable SysUtils ShortTimeFormat Short version of the time to string format
22934: Procedure Dialogs ShowMessage Display a string in a simple dialog with an OK button
22935: Procedure Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
22936: Procedure Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
22937: Keyword Shr Shift an integer value right by a number of bits
22938: Function System Sin The Sine of a number
22939: Type System Single The smallest capacity and precision floating point type
22940: Function System SizeOf Gives the storage byte size of a type or variable
22941: Function System Slice Creates a slice of an array as an Open Array parameter
22942: Type System SmallInt An Integer type supporting values from -32768 to 32767
22943: Function System Sqr Gives the square of a number
22944: Function System Sqrt Gives the square root of a number
22945: Procedure System Str Converts an integer or floating point number to a string
22946: Type System String A data type that holds a string of characters
22947: Function System StringOfChar Creates a string with one character repeated many times
22948: Function SysUtils StringReplace Replace one or more substrings found within a string
22949: Function System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
22950: Function SysUtils StrScan Searches for a specific character in a constant string
22951: Function SysUtils StrToCurr Convert a number string into a currency value
22952: Function SysUtils StrToDate Converts a date string into a TDateTime value
22953: Function SysUtils StrToDateTime Converts a date+time string into a TDateTime value
22954: Function SysUtils StrToFloat Convert a number string into a floating point value
22955: Function SysUtils StrToInt Convert an integer string into an Integer value
22956: Function SysUtils StrToInt64 Convert an integer string into an Int64 value
22957: Function SysUtils StrToInt64Def Convert a string into an Int64 value with default
22958: Function SysUtils StrToIntDef Convert a string into an Integer value with default
22959: Function SysUtils StrToTime Converts a time string into a TDateTime value
22960: Function StrUtils StuffString Replaces a part of one string with another
22961: Function System Succ Increment an ordinal variable
22962: Function Math Sum Return the sum of an array of floating point values
22963:
22964: T
22965: Function Math Tan The Tangent of a number
22966: Type Classes TBits An object that can hold an infinite number of Boolean values
22967: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
22968: Type ConvUtils TConvType Defines a measurement type as used by Convert
22969: Type System TDateTime Data type holding a date and time value
22970: Type System Text Defines a file as a text file
22971: Type System TextFile Declares a file type for storing lines of text
22972: Type SysUtils TFloatFormat Formats for use in floating point number display functions
22973: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
22974: Keyword Then Part of an if statement - starts the true clause
22975: Variable SysUtils ThousandSeparator The character used to display the thousands separator
22976: Keyword ThreadVar Defines variables that are given separate instances per thread
22977: Function SysUtils Time Gives the current time
22978: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
22979: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
22980: Variable SysUtils TimeSeparator The character used to separate display time fields
22981: Function SysUtils TimeToStr Converts a TDateTime time value to a string
22982: Type Classes TList General purpose container of a list of objects
22983: Keyword To Prefixes an incremental for loop target value
22984: Type System TObject The base class type that is ancestor to all other classes
22985: Function DateUtils Tomorrow Gives the date tomorrow
22986: Type Dialogs TOpenDialog Displays a file selection dialog
22987: Type Types TPoint Holds X and Y integer values
22988: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
22989: Type Types TRect Holds rectangle coordinate values
22990: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
22991: Function SysUtils Trim Removes leading and trailing blanks from a string
22992: Function SysUtils TrimLeft Removes leading blanks from a string
22993: Function SysUtils TrimRight Removes trailing blanks from a string
22994: Function System Trunc The integer part of a floating point number
22995: Procedure System Truncate Truncates a file size - removes all data after the current position
22996: Keyword Try Starts code that has error trapping
22997: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
22998: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
22999: Type Classes TStringList Holds a variable length list of strings
23000: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
23001: Type System TThreadFunc Defines the function to be called by BeginThread
23002: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
23003: Keyword Type Defines a new category of variable or process
23004:
23005: U
23006: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
23007: Keyword Unit Defines the start of a unit file - a Delphi module
23008: Keyword Until Ends a Repeat control loop
23009: Function System UpCase Convert a Char value to upper case
23010: Function SysUtils UpperCase Change lower case characters in a string to upper case
23011: Keyword Uses Declares a list of Units to be imported
23012:
23013: V
23014: Procedure System Val Converts number strings to integer and floating point values
23015: Keyword Var Starts the definition of a section of data variables
23016: Type System Variant A variable type that can hold changing data types
23017: Function Variants VarType Gives the current type of a Variant variable
23018: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
23019: Directive Virtual Allows a class method to be overriden in derived classes
```

```
23020:
23021: W
23022: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
23023: Keyword While Repeat statements whilst a continuation condition is met
23024: Type System WideChar Variable type holding a single International character
23025: Function System WideCharToString Copies a null terminated WideChar string to a normal string
23026: Type System WideString A data type that holds a string of WideChars
23027: Keyword With A means of simplifying references to structured variables
23028: Type System Word An integer type supporting values 0 to 65535
23029: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
23030: Procedure System Write Write data to a binary or text file
23031: Procedure System WriteLn Write a complete line of data to a text file
23032:
23033: X
23034: Compiler Directive $X Controls some Pascal extension handling
23035: Keyword Xor Boolean Xor or bitwise Xor of two arguments
23036:
23037: Y
23038: Compiler Directive $Y Determines whether application symbol information is built
23039: Function DateUtils Yesterday Gives the date yesterday
23040:
23041: Z
23042: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
23043:
23044:
23045: SHA1:  maXbox3.exe D0EC95326FE1ABD9D441F137336D00CF4BC77CAB
23046: CRC32: maXbox3.exe D74E54A7
23047: Ref:
23048:     1. writeln(SHA1(Exepath+'\maxbox3.exe'))
23049:     2. shdig: TSHA1Digest;
23050:        shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
23051:                for i:= 0 to 19 do write(BytetoHex(shdig[i]));
23052:
23053:     3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
23054:
23055:   ---- bigbitbox code_cleared_checked----
```