

```

1: ****
2: Constructor Function and Procedure List of maxbox 3.9.9.101
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: 23353344 V3.9.9.101 Nov 2014 To EKON/BASTA/JAX/IBZ/SWS/EU/DT/PASCON
10: *****Now the Funclist*****
11: Funclist Function : 13694 //10766 //10165 (7648)
12: *****Now the Proclist*****
13: Proclist Procedure Size is: 8406 //6792 //6401 4752
14: *****Now Constructors*****
15: Constructlist Constructor Size is: 1372 //995 //
16: def head:max: maxbox7: 28.10.2014 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: 23472! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 23870
22: ASize of EXE: 23353344 (22041600) (16586240) (13511680)
23: SHA1 Hash of maxbox 3.9.9.101: E951E93E67DE0E815D3F7441842994BAEA06F21A
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: TTextLineStyle', '(tlbsLF, tlbsCRLF)
109: Function AdjustLineBreaks(const S: string; Style: TTextLineStyle): 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( : 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 AUusername, 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 Bcd.ToDouble( 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 Binary.ToDouble( ABinary : string; DefValue : Double) : Double;
239: Function BinomialCoeff( N, R : Cardinal) : Float;
240: function BinomialCoefficient(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 BoolToStrl(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 : TIddBytes; const AIIndex : Integer) : Cardinal
283: Function BytesToChar( const AValue : TIddBytes; const AIIndex : Integer) : Char
284: Function BytesToInt64( const AValue : TIddBytes; const AIIndex : Integer) : Int64
285: Function BytesToInteger( const AValue : TIddBytes; const AIIndex : Integer) : Integer
286: Function BytesToIPv6( const AValue : TIddBytes; const AIIndex : Integer) : TIidIpv6Address
287: Function BytesToShort( const AValue : TIddBytes; const AIIndex : Integer) : Short
288: Function BytesToString(ABytes:TIdBytes; AstartIndex:Integer; AMaxCount:Integer): string;
289: Function BytesToStr(const Value : TBytes): String;
290: Function BytesToWord( const AValue : TIddBytes; const AIIndex : 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 CharIsEOF( const AString : string; ACharPos : Integer) : Boolean
327: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : String) : Boolean
328: Function CharLength( S : String; Index : Integer) : Integer
329: Function CharRange( const AMin, AMax : Char) : String
330: function CharsetToIdent(Charset: Longint; var Ident: string): Boolean)
331: Function CharToBin(vchr: Char): String;
332: Function CharNext(lpsz: PChar): PChar; stdcall;
333: Function CharToByteIndex( S : string; Index : Integer) : Integer
334: function CharToByteIndex(const S: string; Index: Integer): Integer)
335: Function CharToByteLen( S : string; MaxLen : Integer) : Integer
336: function CharToByteLen(const S: string; MaxLen: Integer): Integer)
337: Function CharToHex(const APrefix : String; const cc : Char) : shortstring;
338: function CharToHexStr(Value: char): string);
339: function CharToOem(ins, outs: PChar):boolean;
340: function CharToUniCode(Value: Char): string;
341: Function CheckMenuDropdown : Boolean
342: Function CheckMessages : longint
343: Function CheckBox: string;
344: Function CheckOpen( Status : DBIResult) : Boolean
345: Function CheckPassword( const APassword : String) : Boolean
346: Function CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt): SmallInt
347: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
348: function CheckSynchronize(Timeout: Integer): Boolean
349: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
350: Function CheckCom(AComNumber: Integer): Integer;');
351: Function CheckLPT1: string;');
352: function ChrA(const a: byte): char;
353: Function ClassIDToProgID(const ClassID: TGUID): string;
354: Function ClassNameIs(const Name: string): Boolean
355: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
356: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;

```

```

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

```

```

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

```

```

524: Function CreateTwoColorsBrushPattern( Color1, Color2 : TColor ) : TBitmap
525: Function CreateValueBuffer( Length : Integer ) : TValueBuffer
526: Function CreateHexDump( AOwner : TWinControl ) : THexDump
527: Function Csc( const X : Extended ) : Extended
528: Function CscH( const X : Extended ) : Extended
529: function currencyDecimals: Byte
530: function currencyFormat: Byte
531: function currencyString: String
532: Function CurrentProcessId : TIdPID
533: Function CurrentReadBuffer : string
534: Function CurrentThreadId : TIdPID
535: Function CurrentYear : Word
536: Function CurrToBCD( const Curr: Currency; var BCD: TBcd; Precision: Integer; Decimals: Integer): Boolean
537: Function CurrToStr( Value : Currency ) : string;
538: Function CurrToStrF( Value : Currency; FormatSettings : TFormatSettings; Digits: Integer ) : string;
539: Function CurrToStrFS(Value:Currency;Format:TFloatFormat;Digits:Int;const
  FormatSettings:TFormatSettings):string;
540: function CursorToIdent(cursor: Longint; var Ident: string): Boolean;
541: function CursorToString(cursor: TCursor): string;
542: Function CustomSort( SortProc : TLVCompare; lParam : Longint ) : Boolean
543: Function CustomSort( SortProc : TTVCompare; Data : Longint; ARecurse : Boolean ) : Boolean
544: Function CycleToDeg( const Cycles : Extended ) : Extended
545: Function CycleToGrad( const Cycles : Extended ) : Extended
546: Function CycleToRad( const Cycles : Extended ) : Extended
547: Function D2H( N : Longint; A : Byte ) : string
548: Function DarkColor( const Color : TColor; const Pct : Single ) : TColor
549: Function DarkColorChannel( const Channel : Byte; const Pct : Single ) : Byte
550: Function DataLinkDir : string
551: Function DataRequest( Data : OleVariant ) : OleVariant
552: Function DataRequest( Input : OleVariant ) : OleVariant
553: Function DataToRawColumn( ACol : Integer ) : Integer
554: Function Date : TDateTime
555: function Date: TDateTime;
556: Function DateIsNull( const pdtValue : TDateTime; const pdtKind : TdtKind ) : Boolean
557: Function DateOf( const AValue : TDateTime ) : TDateTime
558: function DateSeparator: char;
559: Function DateTimeGMTToHttpStr( const GMTValue : TDateTime ) : String
560: Function DateTimeToFileDate( DateTime : TDateTime ) : Integer
561: function DateTimeToFileDate(DateTime: TDateTime): Integer;
562: Function DateTimeToGmtOffsetStr( ADateTime : TDateTime; SubGMT : Boolean ) : string
563: Function DateTimeToInternetStr( const Value : TDateTime; const AIsGMT : Boolean ) : String
564: Function DateTimeToJulianDate( const AValue : TDateTime ) : Double
565: Function DateTimeToModifiedJulianDate( const AValue : TDateTime ) : Double
566: Function DateTimeToStr( DateTime : TDateTime ) : string;
567: Function DateTimeToStr2( DateTime : TDateTime; FormatSettings : TFormatSettings ) : string;
568: function DatetimeToTimeStamp(DateTime: TDateTime): TTimeStamp
569: Function DateToUnix( const AValue : TDateTime ) : Int64
570: function DateToUnix(D: TDateTime): Int64;
571: Function DateToStr( DateTime : TDateTime ) : string;
572: function DateToStr(const DateTime: TDateTime): string;
573: function DateToStr(D: TDateTime): string;
574: Function DateToStr2( DateTime : TDateTime; FormatSettings : TFormatSettings ) : string;
575: Function DayOf( const AValue : TDateTime ) : Word
576: Function DayOfTheMonth( const AValue : TDateTime ) : Word
577: function DayOfTheMonth(const AValue: TDateTime): Word;
578: Function DayOfTheWeek( const AValue : TDateTime ) : Word
579: Function DayOfTheYear( const AValue : TDateTime ) : Word
580: function DayOfTheYear(const AValue: TDateTime): Word;
581: Function DayOfWeek( DateTime : TDateTime ) : Word
582: function DayOfWeek(const DateTime: TDateTime): Word;
583: Function DayOfWeekStr( DateTime : TDateTime ) : string
584: Function DaysBetween( const ANow, AThen : TDateTime ) : Integer
585: Function DaysInAMonth( const AYear, AMonth : Word ) : Word
586: Function DaysInAYear( const AYear : Word ) : Word
587: Function DaysInMonth( const AValue : TDateTime ) : Word
588: Function DaysInYear( const AValue : TDateTime ) : Word
589: Function DaySpan( const ANow, AThen : TDateTime ) : Double
590: Function DBUseRightToLeftAlignment( AControl : TControl; AField : TField ) : Boolean
591: function DecimalSeparator: char;
592: Function DecLimit( var B : Byte; const Limit : Byte; const Decr : Byte ) : Byte;
593: Function DecLimit1( var B : Shortint; const Limit : Shortint; const Decr : Shortint ) : Shortint;
594: Function DecLimit2( var B : Smallint; const Limit : Smallint; const Decr : Smallint ) : Smallint;
595: Function DecLimit3( var B : Word; const Limit : Word; const Decr : Word ) : Word;
596: Function DecLimit4( var B : Integer; const Limit : Integer; const Decr : Integer ) : Integer;
597: Function DecLimit5( var B : Cardinal; const Limit : Cardinal; const Decr : Cardinal ) : Cardinal;
598: Function DecLimit6( var B : Int64; const Limit : Int64; const Decr : Int64 ) : Int64;
599: Function DecLimitClamp( var B : Byte; const Limit : Byte; const Decr : Byte ) : Byte;
600: Function DecLimitClamp1( var B : Shortint; const Limit : Shortint; const Decr : Shortint ) : Shortint;
601: Function DecLimitClamp2( var B : Smallint; const Limit : Smallint; const Decr : Smallint ) : Smallint;
602: Function DecLimitClamp3( var B : Word; const Limit : Word; const Decr : Word ) : Word;
603: Function DecLimitClamp4( var B : Integer; const Limit : Integer; const Decr : Integer ) : Integer;
604: Function DecLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Decr : Cardinal ) : Cardinal;
605: Function DecLimitClamp6( var B : Int64; const Limit : Int64; const Decr : Int64 ) : Int64;
606: Function DecodeDateFully( DateTime : TDateTime; var Year, Month, Day, DOW : Word ) : Boolean
607: Function DecodeSoundexInt( AValue : Integer ) : string
608: Function DecodeSoundexWord( AValue : Word ) : string
609: Function DefaultAlignment : TAlignment
610: Function DefaultCaption : string
611: Function DefaultColor : TColor

```

```

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

```

```

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

```

```

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

```

```

878: Function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer
879: Function FindNext2(var F: TSearchRec): Integer
880: procedure FindClose2(var F: TSearchRec)
881: Function FINDFIRST : BOOLEAN
882: TJvSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
883:   sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms, sfRecent, sfSendTo,
884:   sfStartMenu, stStartUp, sfTemplates);
884: FFolder: array [TJvSpecialFolder] of Integer =
885:   (CSIDL_BITBUCKET, CSDL_CONTROLS, CSDL_DESKTOP, CSDL_DESKTOPDIRECTORY,
886:    CSDL_DRIVES, CSDL_FONTS, CSDL_NETHOOD, CSDL_NETWORK, CSDL_PERSONAL,
887:    CSDL_PRINTERS, CSDL_PROGRAMS, CSDL_RECENT, CSDL_SENDTO, CSDL_STARTMENU,
888:    CSDL_STARTUP, CSDL_TEMPLATES);
889: Function FindFilesDlg(StartIn: string; SpecialFolder: TJvSpecialFolder; UseFolder: Boolean): Boolean;
890: function Findfirst(const filepath: string; attr: integer): integer;
891: function Findfirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer
892: Function FindFirstNotOf( AFind, AText : String) : Integer
893: Function FindFirstOf( AFind, AText : String) : Integer
894: Function FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState
895: Function FINDINDEXFORFIELDS( const FIELDS : String) : TINDEXDEF
896: Function FindInstanceOf( AClass : TClass; AExact: Boolean; AStartAt : Integer) : Integer
897: Function FINDITEM( VALUE : INTEGER; KIND : TFINDITEMKIND) : TMENUITEM
898: function FindItemId( Id : Integer) : TCollectionItem
899: Function FindKey( const KeyValues : array of const) : Boolean
900: Function FINDLAST : BOOLEAN
901: Function FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl
902: Function FindModuleClass( AClass : TComponentClass) : TComponent
903: Function FindModuleName( const AClass : string) : TComponent
904: Function FINDNEXT : BOOLEAN
905: function FindNext: integer;
906: function FindNext2(var F: TSearchRec): Integer
907: Function FindNextPage( CurPage: TTabSheet; GoForward, CheckTabVisible : Boolean) : TTabSheet
908: Function FindNextToSelect : TTreeNode
909: Function FINDPARAM( const VALUE : String) : TPARAM
910: Function FindParam( const Value : WideString) : TParameter
911: Function FINDPRIOR : BOOLEAN
912: Function FindResource( ModuleHandle : HMODULE; ResourceName, ResourceType : PChar) : TResourceHandle
913: Function FindSession( const SessionName : string) : TSession
914: function FindStringResource(Ident: Integer): string
915: Function FindText( const SearchStr:string;StartPos,Length: Integer; Options: TSearchTypes):Integer
916: Function FindUnusedFileName( const FileName, FileExt, Suffix : AnsiString) : AnsiString
917: function FindVCLWindow(const Pos: TPoint): TWinControl;
918: function FindWindow(C1, C2: PChar): Longint;
919: Function FindinPaths(const fileName,paths: String): String;
920: Function Finger : String
921: Function First : TClass
922: Function First : TComponent
923: Function First : TObject
924: Function FirstDelimiter( const delimiters : string; const Str : String) : integer;
925: Function FirstDelimiter1( const delimiters : WideString; const Str : WideString) : integer;
926: Function FirstInstance( const ATitle : string) : Boolean
927: Function FloatPoint( const X, Y : Float) : TFloatPoint;
928: Function FloatPoint1( const P : TPoint) : TFloatPoint;
929: Function FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) : Boolean
930: Function FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
931: Function FloatRect1( const Rect : TRect) : TFloatRect;
932: Function FloatsEqual( const X, Y : Float) : Boolean
933: Function FloatToBin(const D: Double): string; //doubletohex -> hextobin! in buffer
934: Function FloatToCurr( Value : Extended) : Currency
935: Function FloatToDateTIme( Value : Extended) : TDateTIme
936: Function FloatToStr( Value : Extended) : string;
937: Function FloatToStr(e : Extended) : String;
938: Function FloatToStrF( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer) : string;
939: function FloatToStrF(Value: Extended; Format: TFloatFormat;Precision:Integer; Digits: Integer): string)
940: Function FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings : TFormatSettings) : string;
941: Function FloatToStrFS( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings : TFormatSettings) : string;
942: function FloatToText(BufferArg: PChar; const Value: Extended; ValueType: TFloatValue;Format: TFloatFormat; Precision,Digits: Integer)
943: Function Floor( const X : Extended) : Integer
944: Function FloorInt( Value : Integer; StepSize : Integer) : Integer
945: Function FloorJ( const X : Extended) : Integer
946: Function Flush( const Count : Cardinal) : Boolean
947: Function Flush(var t: Text): Integer
948: function FmtLoadStr(Ident: Integer; const Args: array of const): string
949: function FOCUSED:BOOLEAN
950: Function ForceBackslash( const PathName : string) : string
951: Function ForceDirectories( const Dir : string) : Boolean
952: Function ForceDirectories( Dir : string) : Boolean
953: Function ForceDirectories( Name : string) : Boolean
954: Function ForceInBox( const Point : TPoint; const Box : TRect) : TPoint
955: Function ForceInRange( A, Min, Max : Integer) : Integer
956: Function ForceInRangeR( const A, Min, Max : Double) : Double
957: Function ForEach( AProc : TBucketProc; AInfo : Pointer) : Boolean;
958: Function ForEach1( AEvent : TBucketEvent) : Boolean;
959: Function ForegroundTask: Boolean
960: function Format(const Format: string; const Args: array of const): string;
961: Function FormatBcd( const Format : string; Bcd : TBcd) : string
962: FUNCTION FormatBigInt(s: string): STRING;

```

```

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

```

```

1050: Function GetComPorts: TStringlist;
1051: Function GetCommonAppdataFolder : string
1052: Function GetCommonDesktopdirectoryFolder : string
1053: Function GetCommonFavoritesFolder : string
1054: Function GetCommonFilesFolder : string
1055: Function GetCommonProgramsFolder : string
1056: Function GetCommonStartmenuFolder : string
1057: Function GetCommonStartupFolder : string
1058: Function GetComponent( Owner, Parent : TComponent ) : TComponent
1059: Function GetConnectionRegistryFile( DesignMode : Boolean ) : string
1060: Function GetCookiesFolder : string
1061: Function GetCPUSpeed( var CpuSpeed : TFreqInfo ) : Boolean
1062: Function GetCurrent : TFavoriteLinkItem
1063: Function GetCurrent : TlistItem
1064: Function GetCurrent : TTaskDialogBaseButtonItem
1065: Function GetCurrent : TToolButton
1066: Function GetCurrent : TTreeNode
1067: Function GetCurrent : WideString
1068: Function GetCurrentDir : string
1069: function GetCurrentDir: string)
1070: Function GetCurrentFolder : string
1071: Function GETCURRENTRECORD( BUFFER : PCHAR ) : BOOLEAN
1072: Function GetCurrentProcessId : TidPID
1073: Function GetCurrentThreadHandle : THandle
1074: Function GetCurrentThreadId: LongWord; stdcall;
1075: Function GetCustomHeader( const Name : string ) : String
1076: Function GetDataItem( Value : Pointer ) : Longint
1077: Function GetDataLinkFiles( FileNames : TWideStrings; Directory : string ) : Integer;
1078: Function GetDataLinkFiles1( FileNames : TStrings; Directory : string ) : Integer;
1079: Function GETDATASIZE : INTEGER
1080: Function GetDC(hdwnd: HWND): HDC;
1081: Function GetDefaultFileExt( const MIMEType : string ) : string
1082: Function GetDefaults : Boolean
1083: Function GetDefaultSchemaName : WideString
1084: Function GetDefaultStreamLoader : IStreamLoader
1085: Function GetDesktopDirectoryFolder : string
1086: Function GetDesktopFolder : string
1087: Function GetDFAState( oStates : TList ) : TniRegularExpressionState
1088: Function GetDirectorySize( const Path : string ) : Int64
1089: Function GetDisplayWidth : Integer
1090: Function GetDLLVersion( const DLLName : string; var pdwMajor, pdwMinor : Integer ) : Boolean
1091: Function GetDomainName : string
1092: Function GetDriverRegistryFile( DesignMode : Boolean ) : string
1093: function GetDriveType(rootpath: pchar): cardinal;
1094: Function GetDriveTypeStr( const Drive : Char ) : string
1095: Function GetEnumerator : TFavoriteLinkItemsEnumerator
1096: Function GetEnumerator : TListItemsEnumerator
1097: Function GetEnumerator : TTaskDialogButtonsEnumerator
1098: Function GetEnumerator : TToolBarEnumerator
1099: Function GetEnumerator : TTreeNodesEnumerator
1100: Function GetEnumerator : TWideStringsEnumerator
1101: Function GetEnvVar( const VarName : string ) : string
1102: Function GetEnvironmentVar( const AVariableName : string ) : string
1103: Function GetEnvironmentVariable( const VarName : string ) : string
1104: Function GetEnvironmentVar( const Name : string; var Value : string; Expand : Boolean ) : Boolean
1105: Function GetEnvironmentVars( const Vars : TStrings; Expand : Boolean ) : Boolean
1106: Function getEnvironmentString: string;
1107: Function GetExceptionHandler : TObject
1108: Function GetFavoritesFolder : string
1109: Function GetFieldByName( const Name : string ) : string
1110: Function GetFieldInfo( const Origin : Widestring; var FieldInfo : TFieldInfo ) : Boolean
1111: Function GetFieldValue( ACol : Integer ) : string
1112: Function GetFileAgeCoherence( const FileName : string ) : Boolean
1113: Function GetFileCreation( const FileName : string ) : TfileTime
1114: Function GetFileCreationTime( const Filename : string ) : TDateTime
1115: Function GetFileInfo( const FileName : string ) : TSearchRec
1116: Function GetFileLastAccess( const FileName : string ) : TfileTime
1117: Function GetFileLastWrite( const FileName : string ) : TfileTime
1118: Function GetFileList(FileList: TStringlist; apath: string): TStringlist;
1119: Function GetFileList1(apath: string): TStringlist;
1120: Function GetFileMIMEType( const AFileName : string ) : string
1121: Function GetFileSize( const FileName : string ) : Int64
1122: Function GetFileVersion( AFileName : string ) : Cardinal
1123: Function GetFileVersion( const AFilename : string ) : Cardinal
1124: Function GetFileVersion2(Handle: Integer; x: Integer): Integer; stdcall;
1125: Function GetFileDate(aFile:string; aWithTime:Boolean):string;
1126: Function GetFileCount(adirmask: string): integer; //files count in directory!
1127: Function GetFilterData( Root : PExprNode ) : TExprData
1128: Function getFirstChild : TTreeNode
1129: Function getFirstChild : LongInt
1130: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string ) : string
1131: Function GetFirstNode : TTreeNode
1132: Function GetFontsFolder : string
1133: Function GetFormulaValue( const Formula : string ) : Extended
1134: Function GetFreePageFileMemory : Integer
1135: Function GetFreePhysicalMemory : Integer
1136: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind ) : Integer;
1137: Function GetFreeSystemResources1 : TFreeSystemResources;
1138: Function GetFreeVirtualMemory : Integer

```

```

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

```

```

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

```

```

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

```

```

1400: Function GetWindowsTempFolder : string
1401: Function GetWindowsUserID : string
1402: Function GetWindowsVersion : TWindowsVersion
1403: Function GetWindowsVersionString : string
1404: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1405: Function GMTToLocalDateTime( S : string) : TDateTime
1406: Function GotoKey : Boolean
1407: Function GradToCycle( const Grads : Extended) : Extended
1408: Function GradToDeg( const Grads : Extended) : Extended
1409: Function GradToDeg( const Value : Extended) : Extended;
1410: Function GradToDeg1( const Value : Double) : Double;
1411: Function GradToDeg2( const Value : Single) : Single;
1412: Function GradToRad( const Grads : Extended) : Extended
1413: Function GradToRad( const Value : Extended) : Extended;
1414: Function GradToRad1( const Value : Double) : Double;
1415: Function GradToRad2( const Value : Single) : Single;
1416: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1417: Function GreenComponent( const Color32 : TColor32) : Integer
1418: function GUIDToString(const GUID: TGUID): string
1419: Function HandleAllocated : Boolean
1420: function HandleAllocated: Boolean;
1421: Function HandleRequest : Boolean
1422: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1423: Function HarmonicMean( const X : TDynFloatArray) : Float
1424: Function HasAsParent( Value : TTreeNode) : Boolean
1425: Function HASCHILDDEFS : BOOLEAN
1426: Function HasCurValues : Boolean
1427: Function HasExtendCharacter( const s : UTF8String) : Boolean
1428: Function HasFormat( Format : Word) : Boolean
1429: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1430: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1431: Function HashValue(AStream: TStream): LongWord
1432: Function HashValue(AStream: TStream): Word
1433: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1434: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1435: Function HashValue128( const ASrc: string): T4x4LongWordRecord;
1436: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1437: Function HashValue16( const ASrc : string) : Word;
1438: Function HashValue16Stream( AStream : TStream) : Word;
1439: Function HashValue32( const ASrc : string) : LongWord;
1440: Function HashValue32Stream( AStream : TStream) : LongWord;
1441: Function HasMergeConflicts : Boolean
1442: Function hasMoreTokens : boolean
1443: Function HASPARENT : BOOLEAN
1444: function HasParent: Boolean
1445: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1446: Function HasUTF8BOM( S : TStream) : boolean;
1447: Function HasUTF8BOM1( S : AnsiString) : boolean;
1448: Function Haversine( X : Float) : Float
1449: Function Head( s : string; const subs : string; var tail : string) : string
1450: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1451: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1452: function HELPJUMP(JUMPID:STRING):BOOLEAN
1453: Function HeronianMean( const a, b : Float) : Float
1454: function HexToStr(Value: string): string;
1455: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1456: function HexToBin2(HexNum: string): string;
1457: Function HexToDouble( const Hex : string) : Double
1458: function HexToInt(hexnum: string): LongInt;
1459: function HexToStr(Value: string): string;
1460: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1461: function Hi(vdat: word): byte;
1462: function HiByte(W: Word): Byte)
1463: function High: Int64;
1464: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1465: function HINSTANCE: longword;
1466: function HiWord(l: DWORD): Word)
1467: function HMODULE: longword;
1468: Function HourOf( const AValue : TDateTime) : Word
1469: Function HourOfTheDay( const AValue : TDateTime) : Word
1470: Function HourOfTheMonth( const AValue : TDateTime) : Word
1471: Function HourOfTheWeek( const AValue : TDateTime) : Word
1472: Function HourOfTheYear( const AValue : TDateTime) : Word
1473: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1474: Function HourSpan( const ANow, AThen : TDateTime) : Double
1475: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1476: Function HTMLDecode( const AStr : String) : String
1477: Function HTMLEncode( const AStr : String) : String
1478: Function HTMLEscape( const Str : string) : string
1479: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1480: Function HTTPDecode( const AStr : String) : string
1481: Function HTTPEncode( const AStr : String) : string
1482: Function Hypot( const X, Y : Extended) : Extended
1483: Function IBMax( n1, n2 : Integer) : Integer
1484: Function IBMin( n1, n2 : Integer) : Integer
1485: Function IBRandomString( iLength : Integer) : String
1486: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1487: Function IBStripString( st : String; CharsToStrip : String) : String
1488: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String

```

```

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

```

```

1578: Function IndexOfString( Alist : TStringList; Value : Variant ) : Integer
1579: Function IndyCompareStr( const A1 : string; const A2 : string ) : Integer
1580: Function IndyGetHostName : string
1581: Function IndyInterlockedDecrement( var I : Integer ) : Integer
1582: Function IndyInterlockedExchange( var A : Integer; B : Integer ) : Integer
1583: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer ) : Integer
1584: Function IndyInterlockedIncrement( var I : Integer ) : Integer
1585: Function IndyLowerCase( const A1 : string ) : string
1586: Function IndyStrToBool( const AString : String ) : Boolean
1587: Function IndyUpperCase( const A1 : string ) : string
1588: Function InitCommonControl( CC : Integer ) : Boolean
1589: Function InitTempPath : string
1590: Function InMainThread : boolean
1591: Function InOpArray( W : WideChar; sets : array of WideChar ) : boolean
1592: Function Input : Text
1593: Function InputBox( const ACaption, APrompt, ADefault : string ) : string
1594: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string
1595: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1596: Function InputQuery( const ACaption, APrompt : string; var Value : string ) : Boolean
1597: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean
1598: Function InquireSignal( RtlSigNum : Integer ) : TSignalState
1599: Function InRangeR( const A, Min, Max : Double ) : Boolean
1600: function Insert( Index : Integer ) : TCollectionItem
1601: Function Insert( Index : Integer ) : TComboExItem
1602: Function Insert( Index : Integer ) : THeaderSection
1603: Function Insert( Index : Integer ) : TListItem
1604: Function Insert( Index : Integer ) : TStatusPanel
1605: Function Insert( Index : Integer ) : TWorkArea
1606: Function Insert( Index : LongInt; const Text : string ) : LongInt
1607: Function Insert( Sibling : TTreeNode; const S : string ) : TTreeNode
1608: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM ) : INTEGER
1609: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM ) : INTEGER
1610: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer ) : TTreeNode
1611: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer ) : LongInt
1612: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer ) : TTreeNode
1613: Function Instance : Longint
1614: function InstanceSize : Longint
1615: Function Int(e : Extended) : Extended;
1616: function Int64ToStr(i: Int64): String;
1617: Function IntegerToBcd( const AValue : Integer ) : TBcd
1618: Function Intensity( const Color32 : TColor32 ) : Integer;
1619: Function Intensity( const R, G, B : Single ) : Single;
1620: Function InterestPayment( const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
  FutureValue:Extended; PaymentTime : TPaymentTime ) : Extended
1621: Function InterestRate(NPeriods:Integer;const Payment,PresVal,
  FutureVal:Extended;PaymentTime:TPaymentTime):Extended
1622: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word ) : Boolean
1623: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double ) : Extended
1624: Function InternalUpdateRecord( Tree : TUpdateTree ) : Boolean
1625: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect ) : Boolean
1626: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean
1627: Function IntMibToStr( const Value : string ) : string
1628: Function IntPower( const Base : Extended; const Exponent : Integer ) : Extended
1629: Function IntToBin( Value : cardinal ) : string
1630: Function IntToHex( Value : Integer; Digits : Integer ) : string;
1631: function IntToHex(a: integer; b: integer): string;
1632: Function IntToHex64( Value : Int64; Digits : Integer ) : string;
1633: function IntToHex64(Value: Int64; Digits: Integer): string
1634: Function IntTo3Str( Value : Longint; separator: string ) : string
1635: Function inttobool( aInt : LongInt ) : Boolean
1636: function IntToStr(i: Int64): String;
1637: Function IntToStr64(Value: Int64): string
1638: function IOResult : Integer
1639: Function IPv6AddressToStr(const AValue: TIIPv6Address) : string
1640: function IPAddrToHostName(const IP: string): string;
1641: Function IsAccel(VK: Word; const Str: string): Boolean
1642: Function IsAddressInNetwork( Address : String ) : Boolean
1643: Function IsAdministrator : Boolean
1644: Function IsAlias( const Name : string ) : Boolean
1645: Function IsApplicationRunning( const AClassName, ApplName : string ) : Boolean
1646: Function IsASCII( const AByte : Byte ) : Boolean;
1647: Function IsASCIILDH( const AByte : Byte ) : Boolean;
1648: Function IsAssembly(const FileName: string): Boolean;
1649: Function IsBcdNegative( const Bcd : TBcd ) : Boolean
1650: Function IsBinary(const AChar : Char) : Boolean
1651: function IsConsole: Boolean)
1652: Function IsDelimiter( Delimiters, S : string; Index : Integer ) : Boolean
1653: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean
1654: Function IsDelphiDesignMode : boolean
1655: Function IsDelphiRunning : boolean
1656: Function IsDFAState : boolean
1657: Function IsDirectory( const FileName : string ) : Boolean
1658: Function IsDomain( const S : String ) : Boolean
1659: function IsDragObject(Sender: TObject): Boolean;
1660: Function IsEditing : Boolean
1661: Function ISEMPY : BOOLEAN
1662: Function IsEqual( Value : TParameters ) : Boolean
1663: Function ISEQUAL( VALUE : TPARAMS ) : BOOLEAN
1664: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)

```

```

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

```

```

1754: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1755: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1756: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1757: Function ItemRect( Index : Integer) : TRect
1758: function ITEMRECT(INDEX:INTEGER):TRECT
1759: Function ItemWidth( Index : Integer) : Integer
1760: Function JavahashCode(val: string): Integer;
1761: Function JosephusG(n,k: integer; var graphout: string): integer;
1762: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1763: Function JustName(PathName : string) : string; //in path and ext
1764: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1765: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1766: Function Keepalive : Boolean
1767: Function KeysToShiftState(Keys: Word): TShiftState;
1768: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1769: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1770: Function KeyboardStateToShiftState: TShiftState; overload;
1771: Function Languages : TLanguages
1772: Function Last : TClass
1773: Function Last : TComponent
1774: Function Last : TObject
1775: Function LastDelimiter( Delimiters, S : string) : Integer
1776: function LastDelimiter(const Delimiters: string; const S: string): Integer
1777: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1778: Function Latitude2WGS84(lat: double): double;
1779: Function LCM(m,n:longint):longint;
1780: Function LCMJ( const X, Y : Cardinal) : Cardinal
1781: Function Ldexp( const X : Extended; const P : Integer) : Extended
1782: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1783: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1784: function Length: Integer;
1785: Procedure LetFileList(FileList: TStringlist; apath: string);
1786: function lengthmp3(mp3path: string):integer;
1787: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1788: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1789: function LinesCount(sfilename:string):extended;
1790: function TextfileLineCount(const FileName: string): integer;
1791: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
  L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1792: function LineStart(Buffer, BufPos: PChar): PChar
1793: function LineStart(Buffer, BufPos: PChar): PChar
1794: function ListSeparator: char;
1795: function Ln(x: Extended): Extended;
1796: Function LnXP1( const X : Extended) : Extended
1797: function Lo(vdat: word): byte;
1798: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1799: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1800: Function LoadFileAsString( const FileName : string) : string
1801: Function LoadFromFile( const FileName : string) : TBitmapLoader
1802: function Loadfile(const FileName: TFileName): string;
1803: Function LoadlibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1804: Function LoadPackage(const Name: string): HMODULE
1805: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1806: Function LoadStr( Ident : Integer) : string
1807: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1808: Function LoadWideStr( Ident : Integer) : WideString
1809: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1810: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1811: Function LockServer( fLock : LongBool) : HResult
1812: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1813: Function Log( const X : Extended) : Extended
1814: Function Log10( const X : Extended) : Extended
1815: Function Log2( const X : Extended) : Extended
1816: function LogBase10(X: Float): Float;
1817: Function LogBase2(X: Float): Float;
1818: Function LogBaseN(Base, X: Float): Float;
1819: Function LogN( const Base, X : Extended) : Extended
1820: Function LogOffOS : Boolean
1821: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1822: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1823: Function LongDateFormat: string;
1824: function LongTimeFormat: string;
1825: Function LongWordToFourChar( ACardinal : LongWord) : string
1826: Function LOOKUP(const KEYFIELDS: String;const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1827: Function LookupName( const name : string) : TInAddr
1828: Function LookupService( const service : string) : Integer
1829: function Low: Int64;
1830: Function LowerCase( S : string) : string
1831: Function Lowercase(s : AnyString) : AnyString;
1832: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1833: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1834: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1835: function MainInstance: longword
1836: function MainThreadID: longword
1837: Function Map(x, in_min, in_max, out_min, out_max: integer): integer; //arduino
1838: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1839: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1840: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1841: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer

```

```

1842: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal ) : string
1843: Function MakeFile( const FileName: string): integer');
1844: function MakeLong(A, B: Word): Longint)
1845: Function MakeTempFilename( const APath : String ) : string
1846: Function MakeValidFileName( const Str : string) : string
1847: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1848: function MakeWord(A, B: Byte): Word)
1849: Function MakeYear4digit( Year, Pivot : Integer ) : Integer
1850: Function MapDateTime( const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1851: Function MapValues( Mapping : string; Value : string ) : string
1852: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char ) : string
1853: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer ) : TMaskCharType
1854: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer ) : TMaskDirectives
1855: Function MaskGetFldSeparator( const EditMask : string ) : Integer
1856: Function MaskGetMaskBlank( const EditMask : string ) : Char
1857: Function MaskGetMaskSave( const EditMask : string ) : Boolean
1858: Function MaskIntLiteralToChar( IChar : Char ) : Char
1859: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer ) : Integer
1860: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer ) : Integer
1861: Function MaskString( Mask, Value : String ) : String
1862: Function Match( const sString : string ) : TniRegularExpressionMatchResul
1863: Function Match1( const sString : string; iStart : integer ) : TniRegularExpressionMatchResult
1864: Function Matches( const Filename : string ) : Boolean
1865: Function MatchesMask( const Filename, Mask : string ) : Boolean
1866: Function MatchStr( const AText : string; const AValues : array of string ) : Boolean
1867: Function MatchText( const AText : string; const AValues : array of string ) : Boolean
1868: Function Max( AValueOne, AValueTwo : Integer ) : Integer
1869: function Max(const x,y: Integer): Integer;
1870: Function Max1( const B1, B2 : Shortint ) : Shortint;
1871: Function Max2( const B1, B2 : Smallint ) : Smallint;
1872: Function Max3( const B1, B2 : Word );
1873: function Max3(const x,y,z: Integer): Integer;
1874: Function Max4( const B1, B2 : Integer ) : Integer;
1875: Function Max5( const B1, B2 : Cardinal ) : Cardinal;
1876: Function Max6( const B1, B2 : Int64 ) : Int64;
1877: Function Max64( const AValueOne, AValueTwo : Int64 ) : Int64
1878: Function MaxFloat( const X, Y : Float ) : Float
1879: Function MaxFloatArray( const B : TDynFloatArray ) : Float
1880: Function MaxFloatArrayIndex( const B : TDynFloatArray ) : Integer
1881: function MaxIntValue(const Data: array of Integer):Integer)
1882: Function MaxJ( const B1, B2 : Byte );
1883: function MaxPath: string;
1884: function MaxValue(const Data: array of Double): Double)
1885: Function MaxCalc( const Formula : string ) : Extended //math expression parser
1886: Procedure MaxCalcF( const Formula : string); //out to console memo2
1887: function MD5(const fileName: string): string;
1888: Function Mean( const Data : array of Double ) : Extended
1889: Function Median( const X : TDynFloatArray ) : Float
1890: Function Memory : Pointer
1891: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer ) : Integer
1892: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1893: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1894: Function MessageDlg( const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1895: Function MessageDlgl1(const
  Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1896: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
  Y:Integer):Integer;
1897: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
  Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn ) : Integer;
1898: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
  Longint; X, Y : Integer; const HelpFileName : string ) : Integer;
1899: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
  : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn ) : Integer;
1900: Function MibToId( Mib : string ) : string
1901: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer ) : AnsiString;
1902: Function MidStr( const AText : WideString; const AStart, ACount : Integer ) : WideString;
1903: Function microsecondsToCentimeters(milliseconds: longint): longint; //340m/s speed of sound
1904: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1905: Function MIDIOut( DeviceID : Cardinal ) : IJclMIDIOut
1906: Procedure GetMidiOutputs( const List : TStrings )
1907: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
1908: Procedure GetGEOMap(C_form,apath: string; const Data: string); //C_form: [html/json/xml]
1909: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single ) : TSingleNoteTuningData
1910: Function MIDINoteToStr( Note : TMIDINote ) : string
1911: Function WinMidiOut( DeviceID : Cardinal ) : IJclWinMidiOut
1912: Procedure GetMidiOutputs( const List : TStrings )
1913: Procedure MidiOutCheck( Code : MMResult )
1914: Procedure MidiInCheck( Code : MMResult )
1915: Function MillisecondOf( const AValue : TDateTime ) : Word
1916: Function MillisecondOfTheDay( const AValue : TDateTime ) : LongWord
1917: Function MillisecondOfTheHour( const AValue : TDateTime ) : LongWord
1918: Function MillisecondOfTheMinute( const AValue : TDateTime ) : LongWord
1919: Function MillisecondOfTheMonth( const AValue : TDateTime ) : LongWord
1920: Function MillisecondOfTheSecond( const AValue : TDateTime ) : Word
1921: Function MillisecondOfTheWeek( const AValue : TDateTime ) : LongWord
1922: Function MillisecondOfTheYear( const AValue : TDateTime ) : Int64
1923: Function MillisecondsBetween( const ANow, AThen : TDateTime ) : Int64
1924: Function MillisecondSpan( const ANow, AThen : TDateTime ) : Double
1925: Function milliToDateTIme( Millisecond : Longint ) : TDateTime');

```

```

1926: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
1927: Function millis: int64;
1928: Function Min( AValueOne, AValueTwo : Integer) : Integer
1929: Function Min1( const B1, B2 : Shortint) : Shortint;
1930: Function Min2( const B1, B2 : Smallint) : Smallint;
1931: Function Min3( const B1, B2 : Word) : Word;
1932: Function Min4( const B1, B2 : Integer) : Integer;
1933: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1934: Function Min6( const B1, B2 : Int64) : Int64;
1935: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1936: Function MinClientRect : TRect;
1937: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1938: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1939: Function MinFloat( const X, Y : Float) : Float
1940: Function MinFloatArray( const B : TDynFloatArray) : Float
1941: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1942: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
1943: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1944: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1945: Function MinIntValue( const Data : array of Integer) : Integer
1946: function MinIntValue(const Data: array of Integer):Integer)
1947: Function MinJ( const B1, B2 : Byte) : Byte;
1948: Function MinuteOf( const AValue : TDateTime) : Word
1949: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1950: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1951: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1952: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1953: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1954: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1955: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1956: Function MinValue( const Data : array of Double) : Double
1957: function MinValue(const Data: array of Double): Double)
1958: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1959: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1960: Function ModFloat( const X, Y : Float) : Float
1961: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1962: Function Modify( const Key : string; Value : Integer) : Boolean
1963: Function ModuleCacheID : Cardinal
1964: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1965: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1966: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1967: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1968: Function MonthOf( const AValue : TDateTime) : Word
1969: Function MonthOfTheYear( const AValue : TDateTime) : Word
1970: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1971: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1972: Function MonthStr( Date : TDateTime) : string
1973: Function MouseCoord( X, Y : Integer) : TGridCoord
1974: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1975: Function Movefile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1976: Function MoveNext : Boolean
1977: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1978: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1979: Function Name : string
1980: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
Double;PaymentTime:TPaymentTime):Extended
1981: function NetworkVolume(DriveChar: Char): string
1982: Function NEWBOTTOMLINE : INTEGER
1983: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1984: Function NEWITEM( const ACAPTION : string; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK : TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1985: Function NEWLINE : TMENUITEM
1986: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1987: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left, Right:PExprNode)
1988: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
const ITEMS : array of TC MENUITEM) : TPOPUPMENU
1989: Function NewState( eType : TnRegularExpressionStateType) : TnRegularExpressionState
1990: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
TMenuItem;AENABLED:BOOL): TMENUITEM
1991: Function NEWTOPLINE : INTEGER
1992: Function Next : TIIdAuthWhatsNext
1993: Function NextCharIndex( S : String; Index : Integer) : Integer
1994: Function NextRecordSet : TCustomSQLDataSet
1995: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1996: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQToken) : TSQToken;
1997: Function NextToken : Char
1998: Function nextToken : WideString
1999: function NextToken:Char
2000: Function Norm( const Data : array of Double) : Extended
2001: Function NormalizeAngle( const Angle : Extended) : Extended
2002: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
2003: Function NormalizeRect( const Rect : TRect) : TRect
2004: function NormalizeRect(const Rect: TRect): TRect;
2005: Function Now : TDateTime
2006: function Now2: tDateTime
2007: Function NumProcessThreads : integer
2008: Function NumThreadCount : integer
2009: Function NthDayOfWeek( const AValue : TDateTime) : Word

```

```

2010: Function NtProductType : TNtProductType
2011: Function NtProductTypeString : string
2012: function Null: Variant;
2013: Function NullPoint : TPoint
2014: Function NullRect : TRect
2015: Function Null2Blank(aString:String):String;
2016: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue : Extended; PaymentTime : TPaymentTime ) : Extended
2017: Function NumIP : integer
2018: function Odd(x: Longint): boolean;
2019: Function OffsetFromUTC : TDateTime
2020: Function OffsetPoint( const P, Offset : TPoint ) : TPoint
2021: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer ) : Boolean
2022: function OffsetRect(var Rect : TRect; DX:Integer; DY:Integer): Boolean)
2023: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer ) : Integer
2024: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment ) : Boolean
2025: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency ) : Boolean
2026: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
2027: function OpenBit:Integer
2028: Function OpenDatabase : TDatabase
2029: Function OpenDatabase( const DatabaseName : string ) : TDatabase
2030: Procedure OpenDir(adir: string);
2031: Function OpenGLColorToWinColor( const Red, Green, Blue : Float ) : TColor
2032: Function OpenMap(const Data: string): boolean;
2033: Function OpenMapX(const Data: string): boolean;
2034: Function OpenObject( Value : PChar ) : Boolean;
2035: Function OpenObject1( Value : string ) : Boolean;
2036: Function OpenSession( const SessionName : string ) : TSession
2037: Function OpenVolume( const Drive : Char ) : THandle
2038: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte) : Cardinal
2039: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte ) : LongWord
2040: Function OrdToBinary( const Value : Byte ) : string;
2041: Function OrdToBinary1( const Value : Shortint ) : string;
2042: Function OrdToBinary2( const Value : Smallint ) : string;
2043: Function OrdToBinary3( const Value : Word ) : string;
2044: Function OrdToBinary4( const Value : Integer ) : string;
2045: Function OrdToBinary5( const Value : Cardinal ) : string;
2046: Function OrdToBinary6( const Value : Int64 ) : string;
2047: Function OSCheck( RetVal : Boolean ) : Boolean
2048: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD ) : string
2049: Function OSIdentToString( const OSIdent : DWORD ) : string
2050: Function Output: Text)
2051: Function Overlay( ImageIndex : Integer; Overlay : TOverlay ) : Boolean
2052: Function Owner : TCustomListView
2053: function Owner : TPersistent
2054: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char ) : string
2055: Function PadL( pStr : String; pLth : integer ) : String
2056: Function PadL(s : AnyString;I : longInt) : AnyString;
2057: Function PadLCh( pStr : String; pLth : integer; pChr : char ) : String
2058: Function PadR( pStr : String; pLth : integer ) : String
2059: Function Padr(s : AnyString;I : longInt) : AnyString;
2060: Function PadRCh( pStr : String; pLth : integer; pChr : char ) : String
2061: Function PadString( const AString : String; const ALen : Integer; const AChar : Char ) : String
2062: Function Padz(s : AnyString;I : longInt) : AnyString;
2063: Function PaethPredictor( a, b, c : Byte ) : Byte
2064: Function PARAMBYNAME( const VALUE : String ) : TPARAM
2065: Function ParamByName( const Value : WideString ) : TParameter
2066: Function ParamCount: Integer
2067: Function ParamsEncode( const ASrc : string ) : string
2068: function ParamStr(Index: Integer): string)
2069: Function ParseDate( const DateStr : string ) : TDateTime
2070: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN ) : String
2071: Function ParseSQL( SQL : WideString; DoCreate : Boolean ) : WideString
2072: Function PathAddExtension( const Path, Extension : string ) : string
2073: Function PathAddSeparator( const Path : string ) : string
2074: Function PathAppend( const Path, Append : string ) : string
2075: Function PathBuildRoot( const Drive : Byte ) : string
2076: Function PathCanonicalize( const Path : string ) : string
2077: Function PathCommonPrefix( const Path1, Path2 : string ) : Integer
2078: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer:CmpFmt:TCompactPath):string;
2079: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int:CmpFmt:TCompactPath):string;
2080: Function PathEncode( const ASrc : string ) : string
2081: Function PathExtractFileDirFixed( const S : AnsiString ) : AnsiString
2082: Function PathExtractFileNameNoExt( const Path : string ) : string
2083: Function PathExtractPathDepth( const Path : string; Depth : Integer ) : string
2084: Function PathGetDepth( const Path : string ) : Integer
2085: Function PathGetLongName( const Path : string ) : string
2086: Function PathGetLongName2( Path : string ) : string
2087: Function PathGetShortName( const Path : string ) : string
2088: Function PathIsAbsolute( const Path : string ) : Boolean
2089: Function PathIsChild( const Path, Base : AnsiString ) : Boolean
2090: Function PathIsDiskDevice( const Path : string ) : Boolean
2091: Function PathIsUNC( const Path : string ) : Boolean
2092: Function PathRemoveExtension( const Path : string ) : string
2093: Function PathRemoveSeparator( const Path : string ) : string
2094: Function Payment( Rate:Extended;NPeriods:Int;const PresentVal,
FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2095: Function Peek : Pointer
2096: Function Peek : TObject

```

```

2097: function PERFORM(MSG: CARDINAL; WPARAM, LPARAM: LONGINT): LONGINT
2098: Function PeriodPayment( const Rate: Extended; Period, NPeriods: Integer; const PresentValue, FutureValue : Extended; PaymentTime : TPaymentTime ) : Extended
2099: function Permutation(npr, k: integer): extended;
2100: function PermutationInt(npr, k: integer): Int64;
2101: Function PermutationJ( N, R : Cardinal ) : Float
2102: Function Pi : Extended;
2103: Function PiE : Extended;
2104: Function PixelsToDialogsX( const Pixels : Word ) : Word
2105: Function PixelsToDialogsY( const Pixels : Word ) : Word
2106: Function PlaySound(s: pchar; flag, syncflag: integer): boolean;
2107: Function Point( X, Y : Integer ) : TPoint
2108: function Point(X, Y: Integer): TPoint
2109: Function PointAssign( const X, Y : Integer ) : TPoint
2110: Function PointDist( const P1, P2 : TPoint ) : Double;
2111: function PointDist(const P1,P2: TFloatPoint): Double;
2112: Function PointDist1( const P1, P2 : TFloatPoint ) : Double;
2113: function PointDist2(const P1,P2: TPoint): Double;
2114: Function PointEqual( const P1, P2 : TPoint ) : Boolean
2115: Function PointIsNull( const P : TPoint ) : Boolean
2116: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint ) : Double
2117: Function Poly( const X : Extended; const Coefficients : array of Double ) : Extended
2118: Function PortTCPISOpen(dwPort : Word; ipAddressStr: String): boolean;
2119: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2120: Function Pop : Pointer
2121: Function Pop : TObject
2122: Function PopnStdDev( const Data : array of Double ) : Extended
2123: Function PopnVariance( const Data : array of Double ) : Extended
2124: Function PopulationVariance( const X : TDynFloatArray ) : Float
2125: function Pos(SubStr, S: AnyString): Longint;
2126: Function PosEqual( const Rect : TRect ) : Boolean
2127: Function PosEx( const SubStr, S : string; Offset : Integer ) : Integer
2128: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt ) : Integer
2129: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2130: Function Post1( AURL : string; const ASource : TStrings ) : string;
2131: Function Post2( AURL : string; const ASource : TStream ) : string;
2132: Function Post3( AURL : string; const ASource : TIIDMultiPartFormDataStream ) : string;
2133: Function PostData( const UserData : WideString; const CheckSum : DWORD ) : Boolean
2134: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2135: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2136: Function Power( const Base, Exponent : Extended ) : Extended
2137: Function PowerBig(aval, n:integer): string;
2138: Function PowerIntJ( const X : Float; N : Integer ) : Float;
2139: Function PowerJ( const Base, Exponent : Float ) : Float;
2140: Function PowerOffOS : Boolean
2141: Function PreformatDateString( Ps : string ) : string
2142: Function PresentValue( const Rate:Extended;NPeriods:Int;const Payment,
FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2143: Function PrimeFactors( N : Cardinal ) : TDynCardinalArray
2144: Function Printer : TPrinter
2145: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2146: Function ProcessResponse : TIIDHTTPWhatsNext
2147: Function ProduceContent : string
2148: Function ProduceContentFromStream( Stream : TStream ) : string
2149: Function ProduceContentFromString( const S : string ) : string
2150: Function ProgIDToClassID(const ProgID: string): TGUID;
2151: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString ) : WideString
2152: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString ) : WideString
2153: Function PromptForFileName( var AFileName : string; const AFILTER: string; const ADefaultExt : string;
const ATITLE : string; const AInitialDir : string; SaveDialog : Boolean ) : Boolean
2154: function PromptForFileName(var AFileName: string; const AFILTER: string; const ADefaultExt: string;const
ATITLE: string; const AInitialDir: string; SaveDialog: Boolean): Boolean
2155: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2156: Function PtInRect( const Rect : TRect; const P : TPoint ) : Boolean
2157: function PtInRect(const Rect: TRect; const P: TPoint): Boolean
2158: Function Push( AItem : Pointer ) : Pointer
2159: Function Push( AObject : TObject ) : TObject
2160: Function Put1( AURL : string; const ASource : TStream ) : string;
2161: Function Pythagoras( const X, Y : Extended ) : Extended
2162: Function queryDLLInterface( var queryList : TStringList ) : TStringList
2163: Function queryDLLInterfaceTwo( var queryList : TStringList ) : TStringList
2164: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2165: Function queryPerformanceCounter2(msc: int64): int64;
2166: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2167: //Function QueryPerformanceFrequency(msc: int64): boolean;
2168: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2169: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2170: Procedure QueryPerformanceCounterl(var aC: Int64);
2171: Function QueryPerformanceFrequencyl(var freq: int64): boolean;
2172: Function Quote( const ACommand : String ) : SmallInt
2173: Function QuotedStr( S : string ) : string
2174: Function RadToCycle( const Radians : Extended ) : Extended
2175: Function RadToDeg( const Radians : Extended ) : Extended
2176: Function RadToDeg( const Value : Extended ) : Extended;
2177: Function RadToDeg1( const Value : Double ) : Double;
2178: Function RadToDeg2( const Value : Single ) : Single;
2179: Function RadToGrad( const Radians : Extended ) : Extended
2180: Function RadToGrad( const Value : Extended ) : Extended;
2181: Function RadToGrad1( const Value : Double ) : Double;

```

```

2182: Function RadToGrad2( const Value : Single ) : Single;
2183: Function RandG( Mean, StdDev : Extended ) : Extended;
2184: function Random( const ARange: Integer): Integer;
2185: function random2(a: integer): double;
2186: function RandomE: Extended;
2187: function RandomF: Extended;
2188: Function RandomFrom( const AValues : array of string) : string;
2189: Function RandomRange( const AFrom, ATo : Integer ) : Integer;
2190: function randSeed: longint;
2191: Function RawToDataColumn( ACol : Integer ) : Integer;
2192: Function Read : Char;
2193: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint ) : HResult;
2194: function Read(Buffer:String;Count:LongInt):LongInt;
2195: Function ReadBinaryStream( const Section, Name : string; Value : TStream ) : Integer;
2196: Function ReadBool( const Section, Ident : string; Default : Boolean ) : Boolean;
2197: Function ReadCardinal( const AConvert : boolean ) : Cardinal;
2198: Function ReadChar : Char;
2199: Function ReadClient( var Buffer, Count : Integer ) : Integer;
2200: Function ReadDate( const Section, Name : string; Default : TDateTime ) : TDateTime;
2201: Function ReadDateTime( const Section, Name : string; Default : TDateTime ) : TDateTime;
2202: Function ReadFloat( const Section, Name : string; Default : Double ) : Double;
2203: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:Bool):Int;
2204: Function ReadInteger( const AConvert : boolean ) : Integer;
2205: Function ReadInteger( const Section, Ident : string; Default : Longint ) : Longint;
2206: Function ReadLn : string;
2207: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer ) : string;
2208: function ReadLn(question: string): string;
2209: Function readn: string; //read last line in memo2 - console!
2210: Function ReadLnWait( AFailCount : Integer ) : string;
2211: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2212: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2213: Function ReadSmallInt( const AConvert : boolean ) : SmallInt;
2214: Function ReadString( const ABytes : Integer ) : string;
2215: Function ReadString( const Section, Ident, Default : string ) : string;
2216: Function ReadString( Count : Integer ) : string;
2217: Function ReadTime( const Section, Name : string; Default : TDateTime ) : TDateTime;
2218: Function ReadTimeStampCounter : Int64;
2219: Function RebootOS : Boolean;
2220: Function Receive( ATimeOut : Integer ) : TReplyStatus;
2221: Function ReceiveBuf( var Buf, Count : Integer ) : Integer;
2222: Function ReceiveLength : Integer;
2223: Function ReceiveText : string;
2224: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal;
2225: Function ReceiveSerialText: string;
2226: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word ) : TDateTime;
2227: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSec:Word):TDateTime;
2228: Function RecodeDay( const AValue : TDateTime; const ADay : Word ) : TDateTime;
2229: Function RecodeHour( const AValue : TDateTime; const AHour : Word ) : TDateTime;
2230: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word ) : TDateTime;
2231: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word ) : TDateTime;
2232: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word ) : TDateTime;
2233: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word ) : TDateTime;
2234: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecound:Word):TDateTime;
2235: Function RecodeYear( const AValue : TDateTime; const AYear : Word ) : TDateTime;
2236: Function Reconcile( const Results : OleVariant ) : Boolean;
2237: Function Rect( Left, Top, Right, Bottom : Integer ) : TRect;
2238: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect;
2239: Function Rect2( const ATopLeft, ABottomRight : TPoint ) : TRect;
2240: Function RectAssign( const Left, Top, Right, Bottom : Integer ) : TRect;
2241: Function RectAssignPoints( const TopLeft, BottomRight : TPoint ) : TRect;
2242: Function RectBounds( const Left, Top, Width, Height : Integer ) : TRect;
2243: Function RectCenter( const R : TRect ) : TPoint;
2244: Function RectEqual( const R1, R2 : TRect ) : Boolean;
2245: Function RectHeight( const R : TRect ) : Integer;
2246: Function RectIncludesPoint( const R : TRect; const Pt : TPoint ) : Boolean;
2247: Function RectIncludesRect( const R1, R2 : TRect ) : Boolean;
2248: Function RectIntersection( const R1, R2 : TRect ) : TRect;
2249: Function RectIntersectRect( const R1, R2 : TRect ) : Boolean;
2250: Function RectIsEmpty( const R : TRect ) : Boolean;
2251: Function RectIsNull( const R : TRect ) : Boolean;
2252: Function RectIsSquare( const R : TRect ) : Boolean;
2253: Function RectisValid( const R : TRect ) : Boolean;
2254: Function RectsAreValid( R : array of TRect ) : Boolean;
2255: Function RectUnion( const R1, R2 : TRect ) : TRect;
2256: Function RectWidth( const R : TRect ) : Integer;
2257: Function RedComponent( const Color32 : TColor32 ) : Integer;
2258: Function Refresh : Boolean;
2259: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2260: Function RegisterConversionFamily( const ADescription : string ) : TConvFamily;
2261: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType ) : Boolean;
2262: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType;
2263: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2264: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2265: Function ReleaseHandle : HBITMAP;
2266: Function ReleaseHandle : HENHMETAFILE;
2267: Function ReleaseHandle : HICON;
2268: Function ReleasePalette : HPALETTE;

```

```

2269: Function RemainderFloat( const X, Y : Float ) : Float
2270: Function Remove( AClass : TClass ) : Integer
2271: Function Remove( AComponent : TComponent ) : Integer
2272: Function Remove( AItem : Integer ) : Integer
2273: Function Remove( AItem : Pointer ) : Pointer
2274: Function Remove( AItem : TObject ) : TObject
2275: Function Remove( AObject : TObject ) : Integer
2276: Function RemoveBackslash( const PathName : string ) : string
2277: Function RemoveDF( aString : String ) : String //removes thousand separator
2278: Function RemoveDir( Dir : string ) : Boolean
2279: function RemoveDir(const Dir: string): Boolean
2280: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2281: Function RemoveFileExt( const FileName : string ) : string
2282: Function RemoveHeaderEntry( AHeader, AEntry : string ) : string
2283: Function RenameFile( OldName, NewName : string ) : Boolean
2284: function RenameFile(const OldName: string; const NewName: string): Boolean
2285: Function ReplaceStr( const AText, AFromText, AToText : string ) : string
2286: Function ReplaceText( const AText, AFromText, AToText : string ) : string
2287: Function Replicate(c : char;I : longInt) : String;
2288: Function Request : TWebRequest
2289: Function ResemblesText( const AText, AOther : string ) : Boolean
2290: Function Reset : Boolean
2291: function Reset2(mypath: string):string;
2292: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2293: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor ) : Boolean
2294: Function Response : TWebResponse
2295: Function ResumeSupported : Boolean
2296: Function RETHINKHOTKEYS : BOOLEAN
2297: Function RETHINKLINES : BOOLEAN
2298: Function Retrieve( const MsgNum : Integer; AMsg : TIIdMessage ) : Boolean
2299: Function RetrieveCurrentDir : string
2300: Function RetrieveDeltas( const cdsArray : array of TClientDataset ) : Variant
2301: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIIdMessage ) : Boolean
2302: Function RetrieveMailBoxSize : integer
2303: Function RetrieveMsgSize( const MsgNum : Integer ) : Integer
2304: Function RetrieveProviders( const cdsArray : array of TClientDataset ) : Variant
2305: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStringList ) : boolean
2306: Function ReturnMIMEType( var MediaType, EncType : String ) : Boolean
2307: Function ReverseBits( Value : Byte ) : Byte;
2308: Function ReverseBits1( Value : Shortint ) : Shortint;
2309: Function ReverseBits2( Value : Smallint ) : Smallint;
2310: Function ReverseBits3( Value : Word ) : Word;
2311: Function ReverseBits4( Value : Cardinal ) : Cardinal;
2312: Function ReverseBits4( Value : Integer ) : Integer;
2313: Function ReverseBits5( Value : Int64 ) : Int64;
2314: Function ReverseBytes( Value : Word ) : Word;
2315: Function ReverseBytes1( Value : Smallint ) : Smallint;
2316: Function ReverseBytes2( Value : Integer ) : Integer;
2317: Function ReverseBytes3( Value : Cardinal ) : Cardinal;
2318: Function ReverseBytes4( Value : Int64 ) : Int64;
2319: Function ReverseString( const AText : string ) : string
2320: Function ReversedDNSLookup(const IPAddrs:String;DNSServer:String;Timeout:Int;var HostName:String):Bool;
2321: Function Revert : HRESULT
2322: Function RGB(R,G,B: Byte): TColor;
2323: Function RGB2BGR( const Color : TColor ) : TColor
2324: Function RGB2TColor( R, G, B : Byte ) : TColor
2325: Function RGBToWebColorName( RGB : Integer ) : string
2326: Function RGBToWebColorStr( RGB : Integer ) : string
2327: Function RgbToHtml( Value : TColor ) : string
2328: Function HtmlToRgb(const Value: string): TColor;
2329: Function RightStr( const AStr : String; Len : Integer ) : String
2330: Function RightStr( const AText : AnsiString; const ACount : Integer ) : AnsiString;
2331: Function RightStr( const AText : WideString; const ACount : Integer ) : WideString;
2332: Function ROL( AVal : LongWord; AShift : Byte ) : LongWord
2333: Function ROR( AVal : LongWord; AShift : Byte ) : LongWord
2334: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float ) : TFloatPoint
2335: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2336: Function Round(e : Extended) : Longint;
2337: Function Round64(e: extended): Int64;
2338: Function RoundAt( const Value : string; Position : SmallInt ) : string
2339: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2340: Function RoundTo(const AValue: Extended; const ADigit: TRoundToEXRangeExtended): Extended;'+');
2341: Function SimpleRoundTo(const AValue: Extended; const ADigit: TRoundToRange): Extended;'+');
2342: Function RoundFrequency( const Frequency : Integer ) : Integer
2343: Function RoundInt( Value : Integer; StepSize : Integer ) : Integer
2344: Function RoundPoint( const X, Y : Double ) : TPoint
2345: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double ) : TRect
2346: Function RowCount : Integer
2347: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2348: Function RowRequest( Row : OleVariant; Options : TFetchOptions ) : OleVariant
2349: Function RPos( const ASub, AIn : String; AStart : Integer ) : Integer
2350: Function RRot( const Value : Byte; const Count : TBitRange ) : Byte;
2351: Function RRot1( const Value : Word; const Count : TBitRange ) : Word;
2352: Function RRot2( const Value : Integer; const Count : TBitRange ) : Integer;
2353: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2354: Function RunningProcessesList( const List : TStringList; FullPath : Boolean ) : Boolean
2355: Function RunByteCode(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;'+');
2356: Function RunCompiledScript2(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;'+');

```

```

2357: Function S_AddBackSlash( const ADirName : string ) : string
2358: Function S_AllTrim( const cStr : string ) : string
2359: Function S_AtRepl( const cAT, cStr, cRepl : string ) : string
2360: Function S_Cut( const cStr : string; const iLen : integer ) : string
2361: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer ) : integer
2362: Function S_DirExists( const Adir : string ) : Boolean
2363: Function S_Empty( const cStr : string ) : boolean
2364: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer ) : string
2365: Function S_LargeFontsActive : Boolean
2366: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer ) : Extended
2367: Function S_LTrim( const cStr : string ) : string
2368: Function S_ReadNextTextLineFromStream( stream : TStream ) : string
2369: Function S_RepeatChar( const iLen : integer; const AChar : Char ) : String
2370: Function S_ReplFirst( const cAT, cStr, cRepl : string ) : string
2371: Function S_RoundDecimal( AValue : Extended; APlaces : Integer ) : Extended
2372: Function S_RTrim( const cStr : string ) : string
2373: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer ) : string
2374: //Type TS_ShellExecuteCmd = (seCmdOpen, seCmdPrint, seCmdExplore);
2375: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd ) : string
2376: Function S_Space( const iLen : integer ) : String
2377: Function S_StrBlanks( const cStr : string; const iLen : integer ) : string
2378: Function S_StrBanksCuttoLong( const cStr : string; const iLen : integer ) : string
2379: Function S_StrCRC32( const Text : string ) : LongWORD
2380: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string
2381: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string
2382: Function S_StringtoUTF_8( const AString : string ) : string
2383: Function S_StrBlanks( const cStr : string; const iLen : integer ) : string
2384: function S_StrToReal( const cStr: string; var R: Double): Boolean
2385: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean ) : boolean
2386: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean ) : string
2387: Function S_UTF_8ToString( const AString : string ) : string
2388: Function S_WBox( const AText : string ) : integer
2389: Function SameDate( const A, B : TDateTime ) : Boolean
2390: function SameDate( const A, B : TDateTime ) : Boolean;
2391: Function SameDateTime( const A, B : TDateTime ) : Boolean
2392: function SameDateTime( const A, B : TDateTime ) : Boolean;
2393: Function SamefileName( S1, S2 : string ) : Boolean
2394: Function SameText( S1, S2 : string ) : Boolean
2395: function SameText( const S1: string; const S2: string): Boolean
2396: Function SameTime( const A, B : TDateTime ) : Boolean
2397: function SameTime( const A, B : TDateTime ) : Boolean;
2398: function SameValue( const A, B : Extended; Epsilon: Extended): Boolean //overload;
2399: function SameValue1( const A, B: Double; Epsilon: Double): Boolean //overload;
2400: function SameValue2( const A, B: Single; Epsilon: Single): Boolean //overload;
2401: Function SampleVariance( const X : TDynFloatArray ) : Float
2402: Function Sar( const Value : Shortint; const Count : TBitRange ) : Shortint;
2403: Function Sar1( const Value : Smallint; const Count : TBitRange ) : Smallint;
2404: Function Sar2( const Value : Integer; const Count : TBitRange ) : Integer;
2405: Function SaveToFile( const AFileName : TFileName ) : Boolean
2406: Function SaveAsExcelFile( AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2407: Function SaveAsExcel( aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2408: Function ScanF( const aformat: String; const args: array of const): string;
2409: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2410: Function SearchBuf( Buf:PChar; BufLen:Integer; SelStart,SelLength:Integer;SearchString:String;Options: TStringSearchOptions):PChar
2411: Function SearchBuf2( Buf: string;SelStart,SelLength:Integer; SearchString: String;Options:TStringSearchOptions):Integer
2412: function SearchRecattr: integer;
2413: function SearchRecExcludeAttr: integer;
2414: Function SearchRecfileSize64( const SearchRec : TSearchRec ) : Int64
2415: function SearchRecname: string;
2416: function SearchRecsize: integer;
2417: function SearchRecTime: integer;
2418: Function Sec( const X : Extended ) : Extended
2419: Function Secant( const X : Extended ) : Extended
2420: Function SecH( const X : Extended ) : Extended
2421: Function Second0( const AValue : TDateTime ) : Word
2422: Function Second0ofTheDay( const AValue : TDateTime ) : LongWord
2423: Function Second0ofTheHour( const AValue : TDateTime ) : Word
2424: Function Second0ofTheMinute( const AValue : TDateTime ) : Word
2425: Function Second0ofTheMonth( const AValue : TDateTime ) : LongWord
2426: Function Second0ofTheWeek( const AValue : TDateTime ) : LongWord
2427: Function Second0ofTheYear( const AValue : TDateTime ) : LongWord
2428: Function SecondsBetween( const ANow, AThen : TDateTime ) : Int64
2429: Function SecondSpan( const ANow, AThen : TDateTime ) : Double
2430: Function SectionExists( const Section : string ) : Boolean
2431: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption ) : Boolean
2432: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint ) : HResult
2433: function Seek(Offset:LongInt;Origin:Word):LongInt
2434: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2435: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string; Options : TSelectDirExtOpts; Parent : TWinControl ): Boolean;
2436: Function SelectImage( var AFileName : string; const Extensions, Filter : string ) : Boolean
2437: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2438: Function SendBuf( var Buf, Count : Integer ) : Integer
2439: Function SendCmd( const AOut : string; const AResponse : SmallInt ) : SmallInt;
2440: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt ) : SmallInt;
2441: Function SendKey( AppName : string; Key : Char ) : Boolean
2442: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;

```

```

2443: Function SendStream( AStream : TStream ) : Boolean
2444: Function SendStreamThenDrop( AStream : TStream ) : Boolean
2445: Function SendText( const S : string ) : Integer
2446: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2447: Function SendSerialText(Data: String): cardinal
2448: Function Sent : Boolean
2449: Function ServicesFilePath: string
2450: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer ) : TColor32
2451: Function SetBit( const Value : Byte; const Bit : TBitRange ) : Byte;
2452: Function SetBit1( const Value : Shortint; const Bit : TBitRange ) : Shortint;
2453: Function SetBit2( const Value : Smallint; const Bit : TBitRange ) : Smallint;
2454: Function SetBit3( const Value : Word; const Bit : TBitRange ) : Word;
2455: Function SetBit4( const Value : Cardinal; const Bit : TBitRange ) : Cardinal;
2456: Function SetBit4( const Value : Integer; const Bit : TBitRange ) : Integer;
2457: Function SetBit5( const Value : Int64; const Bit : TBitRange ) : Int64;
2458: Function SetClipboard( NewClipboard : TClipboard ) : TClipboard
2459: Function SetColorBlue( const Color : TColor; const Blue : Byte ) : TColor
2460: Function SetColorFlag( const Color : TColor; const Flag : Byte ) : TColor
2461: Function SetColorGreen( const Color : TColor; const Green : Byte ) : TColor
2462: Function SetColorRed( const Color : TColor; const Red : Byte ) : TColor
2463: Function SetCurrentDir( Dir : string ) : Boolean
2464: function SetCurrentDir(const Dir: string): Boolean
2465: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2466: Function SetDirCreation( const DirName : string; const DateTime : TDateTime ) : Boolean
2467: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime ) : Boolean
2468: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime ) : Boolean
2469: Function SetDisplayResolution( const XRes, YRes : DWORD ) : Longint
2470: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2471: Function SetEnvironmentVar( const Name, Value : string ) : Boolean
2472: Function SetErrorProc( ErrorProc : TSocketErrorProc ) : TSocketErrorProc
2473: Function SetFileCreation( const FileName : string; const DateTime : TDateTime ) : Boolean
2474: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime ) : Boolean
2475: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime ) : Boolean
2476: Function SetTimeStamp( const FileName : string; TimeStamp : Integer ) : Boolean
2477: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2478: Function SetLocalTime( Value : TDateTime ) : boolean
2479: Function SetPrecisionTolerance( NewTolerance : Float ) : Float
2480: Function SetPrinter( NewPrinter : TPrinter ) : TPrinter
2481: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2482: Function SetRGBValue( const Red, Green, Blue : Byte ) : TColor
2483: Function SetSequence( S, Localizar, Substituir : shortstring ) : shortstring
2484: Function SetSize( libNewSize : Longint ) : HResult
2485: Function SetUserObjectFullAccess( hUserObject : THandle ) : Boolean
2486: Function Sgn( const X : Extended ) : Integer
2487: function SHA1(const fileName: string): string;
2488: function SHA256(astr: string; amode: char): string
2489: function SHA512(astr: string; amode: char): string
2490: Function ShareMemoryManager : Boolean
2491: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2492: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2493: Function ShellExecute3(Afilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2494: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE ) : TSHORCUT
2495: Function SHORTCUTTOTEXT( SHORTCUT : TSHORCUT ) : String
2496: function ShortDateFormat: string;
2497: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
  RTL:Bool;EllipsisWidth:Int):WideString
2498: function ShortTimeFormat: string;
2499: function SHOWMODAL:INTEGER
2500: Function ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:
  TColor; BW: Integer;Title:String;BeforeShowModal:TNotifyEvent): TModalResult';
2501: Function
  ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcon:Bool;BefShowModal:TNotifyEvent):TModalResult;
2502: function ShowWindow(C1: HWND; C2: integer): boolean;
//in Dialog
2503: procedure ShowMemory
2504: function ShowMemory2: string;
2505: Function ShutDownOS : Boolean
2506: Function Signe( const X, Y : Extended ) : Extended
2507: Function Sign( const X : Extended ) : Integer
2508: Function Sin(e : Extended) : Extended;
2509: Function sinc( const x : Double) : Double
2510: Function SinJ( X : Float ) : Float
2511: Function Size( const AFileName : String ) : Integer
2512: function Sizeof: Longint;
2513: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle ) : Integer
2514: function SlashSep(const Path, S: String): String
2515: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer ) : Extended
2516: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL ) : DWORD
2517: Function SmallPoint(X, Y: Integer): TSmallPoint)
2518: Function Soundex( const AText : string; ALengt : TSoundexLength ) : string
2519: Function SoundexCompare( const AText, AOther : string; ALengt : TSoundexLength ) : Integer
2520: Function SoundexInt( const AText : string; ALengt : TSoundexIntLength ) : Integer
2521: Function SoundexProc( const AText, AOther : string ) : Boolean
2522: Function SoundexSimilar( const AText, AOther : string; ALengt : TSoundexLength ) : Boolean
2523: Function SoundexWord( const AText : string ) : Word
2524: Function SourcePos : Longint
2525: function SourcePos:LongInt
2526: Function Split0( Str : string; const substr : string ) : TStringList
2527: Procedure SplitNameValuePair( const Line : string; var Name, Value : string )
2528: Function SQLRequiresParams( const SQL : WideString ) : Boolean

```

```

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

```

```

2618: function StrLen( const Str: PChar): Cardinal
2619: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2620: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2621: Function StrLICmp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2622: Function StrLower( Str : PChar) : PChar
2623: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2624: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar
2625: Function StrNew( Str : PChar) : PChar
2626: function StrNew(const Str: PChar): PChar
2627: Function StrNextChar( Str : PChar) : PChar
2628: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2629: Function StrParse( var sString : string; const sDelimiters : string) : string;
2630: Function StrParseL( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2631: Function StrPas( Str : PChar) : string
2632: function StrPas(const Str: PChar): string
2633: Function StrPCopy( Dest : PChar; Source : string) : PChar
2634: function StrPCopy(Dest: PChar; const Source: string): PChar
2635: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2636: Function StrPos( Str1, Str2 : PChar) : PChar
2637: Function StrScan(const Str: PChar; Chr: Char): PChar
2638: Function StrRScan(const Str: PChar; Chr: Char): PChar
2639: Function StrToBcd( const AValue : string) : TBcd
2640: Function StrToBool( S : string) : Boolean
2641: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2642: Function StrToCard( const AStr : String) : Cardinal
2643: Function StrToConv( AText : string; out AType : TConvType) : Double
2644: Function StrToCurr( S : string) : Currency
2645: function StrToCurr(const S: string): Currency
2646: Function StrToCurrDef( S : string; Default : Currency) : Currency
2647: Function StrToDate( S : string) : TDateTime;
2648: function StrToDate(const s: string): TDateTime;
2649: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2650: Function StrToDateDef( S : string; Default : TDate) : TDate;
2651: function StrToDateDef( const S: string): TDateTime
2652: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2653: Function StrToday( const ADay : string) : Byte
2654: Function StrToFloat( S : string) : Extended;
2655: function StrToFloat(s: String): Extended;
2656: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2657: function StrToFloatDef(const S: string; const Default: Extended): Extended
2658: Function StrToFloat( S : string) : Extended;
2659: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2660: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2661: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2662: Function StrToCurr( S : string) : Currency;
2663: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2664: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2665: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2666: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2667: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2668: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2669: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2670: Function StrToDateTime( S : string) : TDateTime;
2671: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2672: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2673: Function StrToFloatRegionalIndependent(AValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2674: Function StrToInt( S : string) : Integer
2675: function StrToInt(s: String): Longint;
2676: Function StrToInt64( S : string) : Int64
2677: function StrToInt64(s: String): int64;
2678: Function StrToInt64Def( S : string; Default : Int64) : Int64
2679: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2680: Function StrToIntDef( S : string; Default : Integer) : Integer
2681: function StrToIntDef(const S: string; Default: Integer): Integer
2682: function StrToIntDef(s: String; def: Longint): Longint;
2683: Function StrToMonth( const AMonth : string) : Byte
2684: Function StrToTime( S : string) : TDateTime;
2685: function StrToTime(const S: string): TDateTime
2686: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2687: Function StrToWord( const Value : String) : Word
2688: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2689: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2690: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2691: Function StrUpper( Str : PChar) : PChar
2692: Function StuffString( const AText : string; AStart, ALen : Cardinal; const ASubText : string) : string
2693: Function Sum( const Data : array of Double) : Extended
2694: Function SumFloatArray( const B : TDynFloatArray) : Float
2695: Function SumInt( const Data : array of Integer) : Integer
2696: Function SumOfSquares( const Data : array of Double) : Extended
2697: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2698: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2699: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2700: Function Supports( CursorOptions : TCursorOptions) : Boolean
2701: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2702: Function SwapWord(w : word): word
2703: Function SwapInt(i : integer): integer
2704: Function SwapLong(L : longint): longint
2705: Function Swap(i : integer): integer
2706: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended

```

```

2707: Function SyncTime : Boolean
2708: Function SysErrorMessage( ErrorCode : Integer ) : string
2709: function SysErrorMessage(ErrorCode: Integer): string)
2710: Function SystemTimeToDate( SystemTime : TSystemTime ) : TDateTime
2711: function SystemTimeToDate( const SystemTime: TSystemTime): TDateTime;
2712: Function SysStringLen( const S: WideString): Integer; stdcall;
2713: Function TabRect( Index : Integer ) : TRect
2714: Function Tan( const X : Extended ) : Extended
2715: Function TaskMessageDlg( const Title,
    Msg:string; DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2716: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx : Longint; DefaultButton : TMsgDlgBtn ) : Integer;
2717: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx : Longint; X, Y : Integer ) : Integer;
2718: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn ) : Integer;
2719: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
    TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string ) : Integer;
2720: Function TaskMessageDlgPosHelp1( const Title, Msg:string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2721: Function TenToY( const Y : Float ) : Float
2722: Function TerminateApp( ProcessID : DWORD; Timeout : Integer ) : TJclTerminateAppResult
2723: Function TerminateTask( Wnd : HWND; Timeout : Integer ) : TJclTerminateAppResult
2724: Function TestBit( const Value : Byte; const Bit : TBitRange ) : Boolean;
2725: Function TestBit2( const Value : Shortint; const Bit : TBitRange ) : Boolean;
2726: Function TestBit3( const Value : Smallint; const Bit : TBitRange ) : Boolean;
2727: Function TestBit4( const Value : Word; const Bit : TBitRange ) : Boolean;
2728: Function TestBit5( const Value : Cardinal; const Bit : TBitRange ) : Boolean;
2729: Function TestBit6( const Value : Integer; const Bit : TBitRange ) : Boolean;
2730: Function TestBit7( const Value : Int64; const Bit : TBitRange ) : Boolean;
2731: Function TestBits( const Value, Mask : Byte ) : Boolean;
2732: Function TestBits1( const Value, Mask : Shortint ) : Boolean;
2733: Function TestBits2( const Value, Mask : Smallint ) : Boolean;
2734: Function TestBits3( const Value, Mask : Word ) : Boolean;
2735: Function TestBits4( const Value, Mask : Cardinal ) : Boolean;
2736: Function TestBits5( const Value, Mask : Integer ) : Boolean;
2737: Function TestBits6( const Value, Mask : Int64 ) : Boolean;
2738: Function TestFDIVInstruction : Boolean
2739: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2740: Function TextExtent( const Text : string ) : TSize
2741: function TextHeight(Text: string): Integer;
2742: Function TextIsSame( const A1 : string; const A2 : string ) : Boolean
2743: Function TextStartsWith( const S, SubS : string ) : Boolean
2744: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2745: Function ConvInteger(i : integer):string;
2746: Function IntegerToText(i : integer):string;
2747: Function TEXTTOSHORTCUT( TEXT : String ) : TSHORCUT
2748: function TextWidth(Text: string): Integer;
2749: Function ThreadCount : integer
2750: function ThousandSeparator: char;
2751: Function Ticks : Cardinal
2752: Function Time : TDateTime
2753: function Time: TDateTime;
2754: function TimeGetTime: int64;
2755: Function TimeOf( const AValue : TDateTime ) : TDateTime
2756: function TimeSeparator: char;
2757: function TimeStampToDate( const TimeStamp: TTimeStamp ): TDateTime
2758: Function TimeStampToMSEcs( TimeStamp : TTimeStamp ) : Comp
2759: function TimeStampToMSEcs( const TimeStamp: TTimeStamp ): Comp)
2760: Function TimeToStr( DateTime : TDateTime ) : string;
2761: function TimeToStr( const DateTime: TDateTime): string;
2762: Function TimeZoneBias : TDateTime
2763: Function ToCommon( const AValue : Double ) : Double
2764: function ToCommon( const AValue: Double): Double;
2765: Function Today : TDateTime
2766: Function ToggleBit( const Value : Byte; const Bit : TBitRange ) : Byte;
2767: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange ) : Shortint;
2768: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange ) : Smallint;
2769: Function ToggleBit3( const Value : Word; const Bit : TBitRange ) : Word;
2770: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange ) : Cardinal;
2771: Function ToggleBit5( const Value : Integer; const Bit : TBitRange ) : Integer;
2772: Function ToggleBit6( const Value : Int64; const Bit : TBitRange ) : Int64;
2773: function TokenComponentIdent:String
2774: Function TokenFloat : Extended
2775: function TokenFloat:Extended
2776: Function TokenInt : Longint
2777: function TokenInt:LongInt
2778: Function TokenString : string
2779: function TokenString:String
2780: Function TokenSymbolIs( const S : string ) : Boolean
2781: function TokenSymbolIs(S:string):Boolean
2782: Function Tomorrow : TDateTime
2783: Function ToRightOf( const pc : TControl; piSpace : Integer ) : Integer
2784: Function ToString : string
2785: Function TotalVariance( const Data : array of Double ) : Extended
2786: Function Trace2( AURL : string ) : string;
2787: Function TrackMenu( Button : TToolButton ) : Boolean
2788: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN ) : INTEGER
2789: Function TranslateURI( const URI : string ) : string

```

```

2790: Function TranslationMatchesLanguages( Exact : Boolean ) : Boolean
2791: Function TransparentStretchBlt( DstDC : HDC; DstX,DstY,DstW,DstH : Integer; SrcDC:HDC;SrcX,SrcY,SrcW,
  SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer ) : Boolean
2792: Function Trim( S : string ) : string;
2793: Function Trim( S : WideString ) : WideString;
2794: Function Trim(s : AnyString) : AnyString;
2795: Function TrimAllOf( ATrim, AText : String ) : String
2796: Function TrimLeft( S : string ) : string;
2797: Function TrimLeft( S : WideString ) : WideString;
2798: function TrimLeft(const S: string): string
2799: Function TrimRight( S : string ) : string;
2800: Function TrimRight( S : WideString ) : WideString;
2801: function TrimRight(const S: string): string
2802: function TrueBoolStrs: array of string
2803: Function Trunc(e : Extended) : Longint;
2804: Function Trunc64(e: extended): Int64;
2805: Function TruncPower( const Base, Exponent : Float ) : Float
2806: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily ) : Boolean;
2807: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily ) : Boolean;
2808: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2809: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime ) : Boolean;
2810: Function TryEncodeDateMonthWeek( const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2811: Function TryEncodeDateTime( const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
  AValue:TDateTime ):Boolean
2812: Function TryEncodeDateWeek( const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2813: Function TryEncodeDayOfWeekInMonth( const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
  AVal:TDateTime ):Bool
2814: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2815: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime ) : Boolean;
2816: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime ) : Boolean;
2817: Function TryLock : Boolean;
2818: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime ) : Boolean;
2819: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
  AMilliSecond : Word; out AResult : TDateTime ) : Boolean;
2820: Function TryStrToBcd( const AValue : string; var Bcd : TBcd ) : Boolean;
2821: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType ) : Boolean;
2822: Function TryStrToDate( S : string; Value : TDateTime ) : Boolean;
2823: Function TryStrToDateTime( S : string; Value : TDateTime ) : Boolean;
2824: Function TryStrToTime( S : string; Value : TDateTime ) : Boolean;
2825: Function TryStrToInt( const S: AnsiString; var I: Integer ) : Boolean;
2826: Function TryStrToInt64( const S: AnsiString; var I: Int64 ) : Boolean;
2827: function TryStrToBool( const S: string; out Value: Boolean): Boolean;
2828: Function TwoByteToWord( AByte1, AByte2 : Byte ) : Word;
2829: Function TwoCharToWord( AChar1, AChar2 : Char ) : Word;
2830: Function TwoToY( const Y : Float ) : Float;
2831: Function UCS4StringToWideString( const S : UCS4String ) : WideString;
2832: Function UIDL( const ADest : TStrings; const AMsgNum : Integer ) : Boolean;
2833: function Unassigned: Variant;
2834: Function UndoLastChange( FollowChange : Boolean ) : Boolean;
2835: function UniCodeToStr( Value: string): string;
2836: Function UnionRect( out Rect : TRect; const R1, R2 : TRect ) : Boolean;
2837: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean;
2838: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal ) : TDateTime;
2839: Function UnixPathToDosPath( const Path : string ) : string;
2840: Function UnixToDateTime( const AValue : Int64 ) : TDateTime;
2841: function UnixToDateTime(U: Int64): TDateTime;
2842: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint ) : HResult;
2843: Function UnlockResource( ResData : HGLOBAL ) : LongBool;
2844: Function UnlockVolume( var Handle : THandle ) : Boolean;
2845: Function UnMaskString( Mask, Value : String ) : String;
2846: function UpCase(ch : Char ) : Char;
2847: Function UpCaseFirst( const AStr : string ) : string;
2848: Function UpCaseFirstWord( const AStr : string ) : string;
2849: Function UpdateAction( Action : TBasicAction ) : Boolean;
2850: Function UpdateKind : TUpdateKind;
2851: Function UPDATESTATUS : TUPDATESTATUS;
2852: Function UpperCase( S : string ) : string;
2853: Function Uppercase(s : AnyString) : AnyString;
2854: Function URLDecode( ASrc : string ) : string;
2855: Function URLEncode( const ASrc : string ) : string;
2856: Function UseRightToLeftAlignment : Boolean;
2857: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment ) : Boolean;
2858: Function UseRightToLeftReading : Boolean;
2859: Function UTF8CharLength( Lead : Char ) : Integer;
2860: Function UTF8CharSize( Lead : Char ) : Integer;
2861: Function UTF8Decode( const S : UTF8String ) : WideString;
2862: Function UTF8Encode( const WS : WideString ) : UTF8String;
2863: Function UTF8LowerCase( const S : UTF8String ) : UTF8String;
2864: Function Utf8ToAnsi( const S : UTF8String ) : string;
2865: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer ) : string;
2866: Function UTF8UpperCase( const S : UTF8String ) : UTF8String;
2867: Function ValidFieldIndex( FieldIndex : Integer ) : Boolean;
2868: Function ValidParentForm(control: TControl): TForm;
2869: Function Value : Variant;
2870: Function ValueExists( const Section, Ident : string ) : Boolean;
2871: Function ValueOf( const Key : string ) : Integer;
2872: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2873: Function VALUEOFKEY( const AKEY : VARIANT ) : VARIANT;
2874: Function VarArrayFromStrings( Strings : TStrings ) : Variant;

```

```

2875: Function VarArrayFromWideStrings( Strings : TWideStrings ) : Variant
2876: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2877: Function VarFMTBcd : TVarType
2878: Function VarFMTBcdCreate1 : Variant;
2879: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word ) : Variant;
2880: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word ) : Variant;
2881: Function VarFMTBcdCreate4( const ABcd : TBcd ) : Variant;
2882: Function Variance( const Data : array of Double ) : Extended
2883: Function VariantAdd2( const V1 : Variant; const V2 : Variant ) : Variant
2884: Function VariantAnd2( const V1 : Variant; const V2 : Variant ) : Variant
2885: Function VariantDiv2( const V1 : Variant; const V2 : Variant ) : Variant
2886: Function VariantGetElement( const V : Variant; il : integer ) : Variant;
2887: Function VariantGetElement1( const V : Variant; il, i2 : integer ) : Variant;
2888: Function VariantGetElement2( const V : Variant; il, i2, i3 : integer ) : Variant;
2889: Function VariantGetElement3( const V : Variant; il, i2, i3, i4 : integer ) : Variant;
2890: Function VariantGetElement4( const V : Variant; il, i2, i3, i4, i5 : integer ) : Variant;
2891: Function VariantMod2( const V1 : Variant; const V2 : Variant ) : Variant
2892: Function VariantMul2( const V1 : Variant; const V2 : Variant ) : Variant
2893: Function VariantNeg( const V1 : Variant ) : Variant
2894: Function VariantNot( const V1 : Variant ) : Variant
2895: Function VariantOr2( const V1 : Variant; const V2 : Variant ) : Variant
2896: Function VariantShl2( const V1 : Variant; const V2 : Variant ) : Variant
2897: Function VariantShr2( const V1 : Variant; const V2 : Variant ) : Variant
2898: Function VariantSub2( const V1 : Variant; const V2 : Variant ) : Variant
2899: Function VariantXor2( const V1 : Variant; const V2 : Variant ) : Variant
2900: function VarIsEmpty(const V: Variant): Boolean;
2901: Function VarIsFMTBcd( const AValue : Variant ) : Boolean;
2902: function VarIsNull(const V: Variant): Boolean;
2903: Function VarToBcd( const AValue : Variant ) : TBcd
2904: function VarType(const V: Variant): TVarType;
2905: Function VarType( const V : Variant ) : TVarType
2906: Function VarAsType( const V : Variant; AVarType : TVarType ) : Variant
2907: Function VarIsType( const V : Variant; AVarType : TVarType ) : Boolean;
2908: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType ) : Boolean;
2909: Function VarIsByRef( const V : Variant ) : Boolean
2910: Function VarIsEmpty( const V : Variant ) : Boolean
2911: Procedure VarCheckEmpty( const V : Variant )
2912: Function VarIsNull( const V : Variant ) : Boolean
2913: Function VarIsClear( const V : Variant ) : Boolean
2914: Function VarIsCustom( const V : Variant ) : Boolean
2915: Function VarIsOrdinal( const V : Variant ) : Boolean
2916: Function VarIsFloat( const V : Variant ) : Boolean
2917: Function VarIsNumeric( const V : Variant ) : Boolean
2918: Function VarIsStr( const V : Variant ) : Boolean
2919: Function VarToStr( const V : Variant ) : string
2920: Function VarToStrDef( const V : Variant; const ADefault : string ) : string
2921: Function VarToWideStr( const V : Variant ) : WideString
2922: Function VarToWideStrDef( const V : Variant; const ADefault : WideString ) : WideString
2923: Function VarToDateTIme( const V : Variant ) : TDateTIme
2924: Function VarFromDateTIme( const DateTIme : TDateTIme ) : Variant
2925: Function VarInRange( const AValue, AMin, AMax : Variant ) : Boolean
2926: Function VarEnsureRange( const AValue, AMin, AMax : Variant ) : Variant
2927: TVariantRelationship', '( vrEqual, vrLessThan, vrGreater Than, vrNotEqual )
2928: Function VarSameValue( const A, B : Variant ) : Boolean
2929: Function VarCompareValue( const A, B : Variant ) : TVariantRelationship
2930: Function VarIsEmptyParam( const V : Variant ) : Boolean
2931: Function VarIsError( const V : Variant; out AResult : HRESULT ) : Boolean;
2932: Function VarIsError1( const V : Variant ) : Boolean;
2933: Function VarAsError( AResult : HRESULT ) : Variant
2934: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant )
2935: Function VarIsArray( const A : Variant ) : Boolean;
2936: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean ) : Boolean;
2937: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType ) : Variant
2938: Function VarArrayOf( const Values : array of Variant ) : Variant
2939: Function VarArrayRef( const A : Variant ) : Variant
2940: Function VarTypeIsValidArrayType( const AVarType : TVarType ) : Boolean
2941: Function VarTypeIsValidElementType( const AVarType : TVarType ) : Boolean
2942: Function VarArrayDimCount( const A : Variant ) : Integer
2943: Function VarArrayLowBound( const A : Variant; Dim : Integer ) : Integer
2944: Function VarArrayHighBound( const A : Variant; Dim : Integer ) : Integer
2945: Function VarArrayLock( const A : Variant ) : __Pointer
2946: Procedure VarArrayUnlock( const A : Variant )
2947: Function VarArrayGet( const A : Variant; const Indices : array of Integer ) : Variant
2948: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer )
2949: Procedure DynArrayToVariant( var V : Variant; const DynArray : __Pointer; TypeInfo : __Pointer )
2950: Procedure DynArrayFromVariant( var DynArray : __Pointer; const V : Variant; TypeInfo : __Pointer )
2951: Function Unassigned : Variant
2952: Function Null : Variant
2953: Function VectorAdd( const V1, V2 : TFloPoint ) : TFloPoint
2954: function VectorAdd(const V1,V2: TFloPoint): TFloPoint;
2955: Function VectorDot( const V1, V2 : TFloPoint ) : Double
2956: function VectorDot(const V1,V2: TFloPoint): Double;
2957: Function VectorLengthSqr( const V : TFloPoint ) : Double
2958: function VectorLengthSqr(const V: TFloPoint): Double;
2959: Function VectorMult( const V : TFloPoint; const s : Double ) : TFloPoint
2960: function VectorMult(const V: TFloPoint; const s: Double): TFloPoint;
2961: Function VectorSubtract( const V1, V2 : TFloPoint ) : TFloPoint
2962: function VectorSubtract(const V1,V2: TFloPoint): TFloPoint;
2963: Function Verify( AUserName : String ) : String

```

```

2964: Function Versine( X : Float ) : Float
2965: function VersionCheck: boolean;
2966: function VersionCheckAct: string;
2967: Function VersionLanguageId( const LangIdRec : TLangIdRec ) : string
2968: Function VersionLanguageName( const LangId : Word ) : string
2969: Function VersionResourceAvailable( const FileName : string ) : Boolean
2970: Function Visible : Boolean
2971: function VolumeID(DriveChar: Char): string
2972: Function WaitFor( const AString : string ) : string
2973: Function WaitFor( const TimeOut : Cardinal ) : TJclWaitResult
2974: Function WaitFor1 : TWaitResult;
2975: Function WaitForData( Timeout : Longint ) : Boolean
2976: Function WebColorNameToColor( WebColorName : string ) : TColor
2977: Function WebColorStrToColor( WebColor : string ) : TColor
2978: Function WebColorToRGB( WebColor : Integer ) : Integer
2979: Function wGet(aURL, afile: string): boolean;
2980: Function wGet2(aURL, afile: string): boolean; //without file open
2981: Function wGetX(aURL, afile: string): boolean;
2982: Function wGetX2(aURL, afile: string): boolean; //without file open
2983: Function WebGet(aURL, afile: string): boolean;
2984: Function WebExists: boolean; //alias to isinternet
2985: Function WeekOf( const AValue : TDateTime ) : Word
2986: Function WeekOfTheMonth( const AValue : TDateTime ) : Word;
2987: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word ) : Word;
2988: Function WeekOfTheYear( const AValue : TDateTime ) : Word;
2989: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word ) : Word;
2990: Function WeeksBetween( const ANow, AThen : TDateTime ) : Integer
2991: Function WeeksInAYear( const AYear : Word ) : Word
2992: Function WeeksInYear( const AValue : TDateTime ) : Word
2993: Function WeekSpan( const ANow, AThen : TDateTime ) : Double
2994: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineStyle ) : WideString
2995: Function WideCat( const x, y : WideString ) : WideString
2996: Function WideCompareStr( S1, S2 : WideString ) : Integer
2997: function WideCompareStr(const S1: WideString; const S2: WideString): Integer
2998: Function WideCompareText( S1, S2 : WideString ) : Integer
2999: function WideCompareText(const S1: WideString; const S2: WideString): Integer
3000: Function WideCopy( const src : WideString; index, count : Integer ) : WideString
3001: Function WideDequotedStr( const S : WideString; AQuote : WideChar ) : WideString
3002: Function WideEqual( const x, y : WideString ) : Boolean
3003: function WideFormat(const Format: WideString; const Args: array of const): WideString)
3004: Function WideGreater( const x, y : WideString ) : Boolean
3005: Function WideLength( const src : WideString ) : Integer
3006: Function WideLess( const x, y : WideString ) : Boolean
3007: Function WideLowerCase( S : WideString ) : WideString
3008: function WidLowerCase(const S: WideString): WideString
3009: Function WidePos( const src, sub : WideString ) : Integer
3010: Function WideQuotedStr( const S : WideString; Quote : WideChar ) : WideString
3011: Function WideReplaceStr( const AText, AFromText, AToText : WideString ) : WideString
3012: Function WideReplaceText( const AText, AFromText, AToText : WideString ) : WideString
3013: Function WideSameStr( S1, S2 : WideString ) : Boolean
3014: function WideSameStr(const S1: WideString; const S2: WideString): Boolean
3015: Function WideSameText( S1, S2 : WideString ) : Boolean
3016: function WideSameText( const S1: WideString; const S2: WideString): Boolean
3017: Function WideStringReplace(const S,OldPattern, NewPattern: WideString; Flags: TReplaceFlags): WideString
3018: Function WideStringToUCS4String( const S : WideString ) : UCS4String
3019: Function WideUpperCase( S : WideString ) : WideString
3020: Function Win32BackupFile( const FileName : string; Move : Boolean ) : Boolean
3021: function Win32Check(RetVal: boolean): boolean
3022: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
3023: Function Win32RestoreFile( const FileName : string ) : Boolean
3024: Function Win32Type : TIIdWin32Type
3025: Function WinColor( const Color32 : TColor32 ) : TColor
3026: function winexec(FileName: pchar; showCmd: integer): integer;
3027: Function WinExec32( const Cmd : string; const CmdShow : Integer ) : Boolean
3028: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer ) : Cardinal
3029: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer ) : Boolean
3030: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64 ) : Boolean
3031: Function WithinPastMilliseconds( const ANow, AThen : TDateTime; const AMilliseconds : Int64 ) : Boolean
3032: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64 ) : Boolean
3033: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer ) : Boolean
3034: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASconds : Int64 ) : Boolean
3035: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer ) : Boolean
3036: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer ) : Boolean
3037: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar ) : DWORD
3038: Function WordToStr( const Value : Word ) : String
3039: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint ) : Boolean
3040: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string ) : Boolean
3041: Procedure GetWordGridFormatValues( Proc : TGetStrProc )
3042: Function WorkArea : Integer
3043: Function WrapText( Line : string; MaxCol : Integer ) : string;
3044: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer ) : string;
3045: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint ) : HResult
3046: function Write(Buffer:String;Count:LongInt):LongInt
3047: Function WriteClient( var Buffer, Count : Integer ) : Integer
3048: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean ) : Cardinal
3049: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string ) : Boolean
3050: Function WriteString( const AString : string ) : Boolean
3051: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
3052: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list ) : Integer

```

```

3053: Function wsprintf( Output : PChar; Format : PChar) : Integer
3054: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
3055: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
3056: Function XorDecode( const Key, Source : string) : string
3057: Function XorEncode( const Key, Source : string) : string
3058: Function XorString( const Key, Src : ShortString) : ShortString
3059: Function Yield : Bool
3060: Function YearOf( const AValue : TDateTime) : Word
3061: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3062: Function YearSpan( const ANow, AThen : TDateTime) : Double
3063: Function Yesterday : TDateTime
3064: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3065: Function( const Name : string; Proc : TUUserFunction)
3066: Function using Special_Scholz from 3.8.5.0
3067: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3068: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3069: Function FloatToTime2Dec(value:Extended):Extended;
3070: Function MinToStd(value:Extended):Extended;
3071: Function MinToStdAsString(value:Extended):String;
3072: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3073: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3074: Function Round2Dec (zahl:Extended):Extended;
3075: Function GetAngle(x,y:Extended):Double;
3076: Function AddAngle(a1,a2:Double):Double;
3077:
3078: ****
3079: unit UPSI_StText;
3080: ****
3081: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3082: Function TextFileSize( var F : TextFile) : LongInt
3083: Function TextPos( var F : TextFile) : LongInt
3084: Function TextFlush( var F : TextFile) : Boolean
3085:
3086: ****
3087: from JvVCLUtils;
3088: ****
3089: { Windows resources (bitmaps and icons) VCL-oriented routines }
3090: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3091: procedure DrawBitmapRectTransparent(Dest: TCanvas; DstX,
  DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparentColor:TColor);
3092: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
  Bitmap: TBitmap; TransparentColor: TColor);
3093: function MakeBitmap(ResID: PChar): TBitmap;
3094: function MakeBitmapID(ResID: Word): TBitmap;
3095: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3096: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3097: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3098: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
  HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3099: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3100: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3101: function AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3102: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3103: {$IFDEF WIN32}
3104: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
  X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3105: {$ENDIF}
3106: function MakeIcon(ResID: PChar): TIcon;
3107: function MakeIconID(ResID: Word): TIcon;
3108: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3109: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3110: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3111: {$IFDEF WIN32}
3112: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3113: {$ENDIF}
3114: { Service routines }
3115: procedure NotImplemented;
3116: procedure ResourceNotFound(ResID: PChar);
3117: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3118: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3119: function PaletteColor(Color: TColor): Longint;
3120: function WidthOf(R: TRect): Integer;
3121: function HeightOf(R: TRect): Integer;
3122: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3123: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3124: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3125: procedure Delay(MSecs: Longint);
3126: procedure DeleteLine(StrList: TStringList; SearchPattern: String);
3127: procedure CenterControl(Control: TControl);
3128: Function PaletteEntries( Palette : HPALETTE ) : Integer
3129: Function WindowClassName( Wnd : HWND ) : string
3130: Function ScreenWorkArea : TRect
3131: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3132: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3133: Procedure ActivateWindow( Wnd : HWND )
3134: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3135: Procedure CenterWindow( Wnd : HWND )
3136: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3137: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3138: Function DialogsToPixelsX( Dlgs : Word ) : Word
3139: Function DialogsToPixelsY( Dlgs : Word ) : Word

```

```

3140: Function PixelsToDialogsX( Pixs : Word) : Word
3141: Function PixelsToDialogsY( Pixs : Word) : Word
3142: {$IFDEF WIN32}
3143: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3144: function MakeVariant(const Values: array of Variant): Variant;
3145: {$ENDIF}
3146: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3147: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3148: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3149: {$IFDEF CBUILDERR}
3150: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3151: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3152: {$ELSE}
3153: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3154: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3155: {$ENDIF CBUILDERR}
3156: function IsForegroundTask: Boolean;
3157: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3158: function GetAveCharSize(Canvas: TCanvas): TPoint;
3159: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3160: procedure FreeUnusedOle;
3161: procedure Beep;
3162: function GetWindowsVersionJ: string;
3163: function LoadDLL(const LibName: string): THandle;
3164: function RegisterServer(const ModuleName: string): Boolean;
3165: {$IFNDEF WIN32}
3166: function IsLibrary: Boolean;
3167: {$ENDIF}
3168: { Gradient filling routine }
3169: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3170: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction: TFillDirection; Colors: Byte);
3171: { String routines }
3172: function GetEnvVar(const VarName: string): string;
3173: function AnsiUpperFirstChar(const S: string): string;
3174: function StringToPChar(var S: string): PChar;
3175: function StrAlloc(const S: string): PChar;
3176: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
3177: function DropT(const S: string): string;
3178: { Memory routines }
3179: function AllocMemo(Size: Longint): Pointer;
3180: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3181: procedure FreeMemo(var fpBlock: Pointer);
3182: function GetMemoSize(fpBlock: Pointer): Longint;
3183: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3184: {$IFNDEF COMPILER5_UP}
3185: procedure FreeAndNil(var Obj);
3186: {$ENDIF}
3187: // from PNGLoader
3188: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3189: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3190: Procedure TransformLOC02RGB( Image : TLinearBitmap)
3191: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3192: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style : TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect //TButtons
3193: Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3194: Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3195: AddConstantN('CTL3D_ALL','LongWord').SetUInt($FFFF);
3196: //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3197: //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3198: //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3199: Procedure SetImeMode( hWnd : HWnd; Mode : TImeMode)
3200: Procedure SetIMEName( Name : TIMEName)
3201: Function Win32NLSEnableIME( hWnd : HWnd; Enable : Boolean) : Boolean
3202: Function Imm32GetContext( hWnd : HWnd) : HIMC
3203: Function Imm32ReleaseContext( hWnd : HWnd; himc : HIMC) : Boolean
3204: Function Imm32GetConversionStatus( himc : HIMC; var Conversion, Sentence : longword) : Boolean
3205: Function Imm32SetConversionStatus( himc : HIMC; Conversion, Sentence : longword) : Boolean
3206: Function Imm32SetOpenStatus( himc : HIMC; fOpen : Boolean) : Boolean
3207: // Function Imm32SetCompositionWindow( himc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3208: //Function Imm32SetCompositionFont( himc : HIMC; lpLogFont : PLOGFONTA) : Boolean
3209: Function Imm32GetCompositionString(himc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3210: Function Imm32IsIME( hkl : longword) : Boolean
3211: Function Imm32NotifyIME( himc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3212: Procedure DragDone( Drop : Boolean)
3213:
3214:
3215: //*****added from jvjclutils
3216: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3217: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3218: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3219: function IsPositiveResult(Value: TModalResult): Boolean;
3220: function IsNegativeResult(Value: TModalResult): Boolean;
3221: function IsAbortResult(const Value: TModalResult): Boolean;
3222: function StripAllFromResult(const Value: TModalResult): TModalResult;
3223: // returns either BrightColor or DarkColor depending on the luminance of AColor
3224: // This function gives the same result (AFAIK) as the function used in Windows to
3225: // calculate the desktop icon text color based on the desktop background color
3226: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor);

```

```

3227: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3228:
3229: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3230:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3231:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3232:   var LinkName: string; Scale: Integer = 100); overload;
3233: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3234:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3235:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3236:   var LinkName: string; Scale: Integer = 100); overload;
3237: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3238:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3239: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3240:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3241:   Scale: Integer = 100): string;
3242: function HTMLPlainText(const Text: string): string;
3243: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3244:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3245: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3246:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3247: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3248: function HTMLPrepareText(const Text: string): string;
3249:
3250: ***** uPSI_JvAppUtils;
3251: Function GetDefaultSection( Component : TComponent ) : string
3252: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3253: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string )
3254: Function GetDefaultIniName : string
3255: //OnGetDefaultIniName , 'TOnGetDefaultIniName';
3256: Function GetDefaultIniRegKey : string
3257: Function FindForm( FormClass : TFormClass ) : TForm
3258: Function FindShowForm( FormClass : TFormClass; const Caption : string ) : TForm
3259: Function ShowDialog( FormClass : TFormClass ) : Boolean
3260: //Function InstantiateForm( FormClass : TFormClass; var Reference ) : TForm
3261: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3262: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3263: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3264: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3265: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3266: Function GetUniqueFileNameInDir( const Path, FileNameMask : string ) : string
3267: Function StrToInStr( const Str : string ) : string
3268: Function IniStrToStr( const Str : string ) : string
3269: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string ) : string
3270: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string )
3271: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint ) : Longint
3272: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint )
3273: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean ) : Boolean
3274: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean )
3275: Procedure IniReadSections( IniFile : TObject; Strings : TStrings )
3276: Procedure IniEraseSection( IniFile : TObject; const Section : string )
3277: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string )
3278: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint )
3279: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint )
3280: Procedure AppTaskbarIcons( AppOnly : Boolean )
3281: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string )
3282: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string )
3283: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject )
3284: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject )
3285: ***** uPSI_JvDBUtils;
3286: Function CreateLocate( DataSet : TDataSet ) : TJvLocateObject
3287: Function IsDataSetEmpty( DataSet : TDataSet ) : Boolean
3288: Procedure RefreshQuery( Query : TDataSet )
3289: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3290: Function DataSetSectionName( DataSet : TDataSet ) : string
3291: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string )
3292: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3293: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
Options: TLocateOptions) : Boolean
3294: Procedure SaveFields( DataSet : TDataSet; IniFile : TIIniFile )
3295: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIIniFile; RestoreVisible : Boolean )
3296: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean )
3297: Function ConfirmDelete : Boolean
3298: Procedure ConfirmDataSetCancel( DataSet : TDataSet )
3299: Procedure CheckRequiredField( Field : TField )
3300: Procedure CheckRequiredFields( const Fields : array of TField )
3301: Function DateToSQL( Value : TDateTime ) : string
3302: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string ) : string
3303: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string ) : string
3304: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
HighEmpty:Double;Inclusive:Bool):string
3305: Function StrMaskSQL( const Value : string ) : string
3306: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3307: Function FormatAnsSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3308: Procedure _DBError( const Msg : string )
3309: Const ('TrueExpr', 'String ''0=0
3310: Const ('sdfStandard16','String ''''''mm''/''dd''/''yyyy'''')
3311: Const ('sdfStandard32','String ''''''dd/mm/yyyy'''')
3312: Const ('sdfOracle','String ''TO_DATE(''''dd/mm/yyyy'''', ''DD/MM/YYYY'''')
3313: Const ('sdfInterbase','String ''CAST(''mm''/''dd''/''yyyy''' AS DATE)'')
```

```

3314: Const ('sdfMSSQL','String ''CONVERT(datetime, ''''mm''/''dd''/''yyyy'''', 103)')
3315: AddTypeS('Largeint', 'Longint'
3316: addTypeS('TIEException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3317:   'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3318:   'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3319:   'erVersionError, ErDivideByZero, ErMathError, erCouldNotCallProc, erOutOfRecordRange, '+
3320:   'erOutOfMemory, erException, erNullPointerException, erNullVariantError, erInterfaceNotSupported, erServerError);
3321: (*-----*)
3322: procedure SJRegister_JclIniFiles(CL: TPSPPascalCompiler);
3323: begin
3324:   Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3325:   Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3326:   Function JIniReadString( const FileName, Section, Line : string) : string
3327:   Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3328:   Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3329:   Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3330:   Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3331:   Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3332: end;
3333:
3334: (* === compile-time registration functions === *)
3335: (*-----*)
3336: procedure SJRegister_JclDateTime(CL: TPSPPascalCompiler);
3337: begin
3338:   'UnixTimeStart', 'LongInt' ( 25569 );
3339:   Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3340:   Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3341:   Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3342:   Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3343:   Function CenturyOfDate( const DateTime : TDateTime) : Integer
3344:   Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3345:   Function DayOfDate( const DateTime : TDateTime) : Integer
3346:   Function MonthOfDate( const DateTime : TDateTime) : Integer
3347:   Function YearOfDate( const DateTime : TDateTime) : Integer
3348:   Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3349:   Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3350:   Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3351:   Function HourOfTime( const DateTime : TDateTime) : Integer
3352:   Function MinuteOfTime( const DateTime : TDateTime) : Integer
3353:   Function SecondOfTime( const DateTime : TDateTime) : Integer
3354:   Function GetISOYearNumberOfDays( const Year : Word) : Word
3355:   Function IsISOLongYear( const Year : Word) : Boolean;
3356:   Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3357:   Function ISODayOfWeek( const DateTime : TDateTime) : Word
3358:   Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3359:   Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3360:   Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3361:   Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3362:   Function JIsLeapYear( const Year : Integer) : Boolean;
3363:   Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3364:   Function JDaysInMonth( const DateTime : TDateTime) : Integer
3365:   Function Make4DigitYear( Year, Pivot : Integer) : Integer
3366:   Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3367:   Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3368:   Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3369:   Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3370:   Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3371:   Function HoursToMSEcs( Hours : Integer) : Integer
3372:   Function MinutesToMSEcs( Minutes : Integer) : Integer
3373:   Function SecondsToMSEcs( Seconds : Integer) : Integer
3374:   Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3375:   Function TimeOfDateTimeToMSEcs( DateTime : TDateTime) : Integer
3376:   Function DateToLocalDateTime( DateTime : TDateTime) : TDateTime
3377:   Function LocalDateTimeToDate( DateTime : TDateTime) : TDateTime
3378:   Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3379:   Function JDATimeToFileTime( DateTime : TDateTime) : TFileTime
3380:   Function JDATimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3381:   Procedure DateToLocalTime( DateTime : TDateTime; var SysTime : TSystemTime);
3382:   Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3383:   Function DosDateTimeToDate( const DosTime : TDosDateTime) : TDateTime
3384:   Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3385:   Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3386:   Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3387:   Function DosDateTimeToStr( DateTime : Integer) : string
3388:   Function JFileTimeToDate( const FileTime : TFileTime) : TDateTime
3389:   Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3390:   Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3391:   Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3392:   Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3393:   Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3394:   Function FileTimeToStr( const FileTime : TFileTime) : string
3395:   Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3396:   Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3397:   Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3398:   Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3399:   Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3400:   Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3401:   Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3402:   TJclUnixTime32', 'Longword

```

```

3403: Function JDateTimeToUnixTime( DateTime : TDateTime ) : TJclUnixTime32
3404: Function JUnixTimeToDate( const UnixTime : TJclUnixTime32 ) : TDateTime
3405: Function FileTimeToUnixTime( const AValue : TFileTime ) : TJclUnixTime32
3406: Function UnixTimeToFileTime( const AValue : TJclUnixTime32 ) : TFileTime
3407: Function JNullStamp : TTimeStamp
3408: Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Int64
3409: Function JEQualTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Boolean
3410: Function JIsNullTimeStamp( const Stamp : TTimeStamp ) : Boolean
3411: Function TimeStampDOW( const Stamp : TTimeStamp ) : Integer
3412: Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3413: Function FirstWeekDay1( const Year, Month : Integer ) : Integer;
3414: Function LastWeekDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3415: Function LastWeekDay1( const Year, Month : Integer ) : Integer;
3416: Function IndexedWeekDay( const Year, Month : Integer; Index : Integer ) : Integer
3417: Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3418: Function FirstWeekendDay1( const Year, Month : Integer ) : Integer;
3419: Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3420: Function LastWeekendDay1( const Year, Month : Integer ) : Integer;
3421: Function IndexedDayOfWeek( const Year, Month : Integer; Index : Integer ) : Integer
3422: Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer ) : Integer
3423: Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer ) : Integer
3424: Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer ) : Integer
3425: FindClass('TOBJECT'), 'EJclDateModelError'
3426: end;
3427:
3428: procedure SIRегистre_JclMiscel2(CL: TPSpascalCompiler);
3429: begin
3430:   Function SetDisplayResolution( const XRes, YRes : DWORD ) : Longint
3431:   Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string ) : Boolean
3432:   Function WinExec32( const Cmd : string; const CmdShow : Integer ) : Boolean
3433:   Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer ) : Cardinal
3434:   Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3435:     TJclKillLevel', '( k1Normal, k1NoSignal, k1TimeOut )
3436:   Function ExitWindows( ExitCode : Cardinal ) : Boolean
3437:   Function LogOffFOS( KillLevel : TJclKillLevel ) : Boolean
3438:   Function PowerOff( KillLevel : TJclKillLevel ) : Boolean
3439:   Function ShutDownOS( KillLevel : TJclKillLevel ) : Boolean
3440:   Function RebootOS( KillLevel : TJclKillLevel ) : Boolean
3441:   Function HibernateOS( Force, DisableWakeEvents : Boolean ) : Boolean
3442:   Function SuspendOS( Force, DisableWakeEvents : Boolean ) : Boolean
3443:   Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean ):Boolean;
3444:   Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool ):Bool;
3445:   Function AbortShutDown : Boolean;
3446:   Function AbortShutDown1( const MachineName : string ) : Boolean;
3447:     TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3448:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3449:   Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3450:   FindClass('TOBJECT'), 'EJclCreateProcessError
3451:   Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string )
3452:   Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
      Environment:PChar);
3453:   // with Add(EJclCreateProcessError) do
3454: end;
3455:
3456:
3457: procedure SIRегистre_JclAnsiStrings(CL: TPSpascalCompiler);
3458: begin
3459:   // 'AnsiSigns','Set').SetSet(['-', '+']);
3460:   'C1_UPPER', 'LongWord( $0001 );
3461:   'C1_LOWER', 'LongWord( $0002 );
3462:   'C1_DIGIT', 'LongWord').SetUInt( $0004 );
3463:   'C1_SPACE', 'LongWord').SetUInt( $0008 );
3464:   'C1_PUNCT', 'LongWord').SetUInt( $0010 );
3465:   'C1_CTRNL', 'LongWord').SetUInt( $0020 );
3466:   'C1_BLANK', 'LongWord').SetUInt( $0040 );
3467:   'C1_XDIGIT', 'LongWord').SetUInt( $0080 );
3468:   'C1_ALPHA', 'LongWord').SetUInt( $0100 );
3469:   AnsiChar', 'Char
3470:   Function StrIsAlpha( const S : AnsiString ) : Boolean
3471:   Function StrIsAlphaNum( const S : AnsiString ) : Boolean
3472:   Function StrIsAlphaNumUnderscore( const S : AnsiString ) : Boolean
3473:   Function StrContainsChars( const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean ) : Boolean
3474:   Function StrConsistsOfNumberChars( const S : AnsiString ) : Boolean
3475:   Function StrIsDigit( const S : AnsiString ) : Boolean
3476:   Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet ) : Boolean
3477:   Function StrSame( const S1, S2 : AnsiString ) : Boolean
3478:   //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar ) : AnsiString
3479:   Function StrCharPosLower( const S : AnsiString; CharPos : Integer ) : AnsiString
3480:   Function StrCharPosUpper( const S : AnsiString; CharPos : Integer ) : AnsiString
3481:   Function StrDoubleQuote( const S : AnsiString ) : AnsiString
3482:   Function StrEnsureNoPrefix( const Prefix, Text : AnsiString ) : AnsiString
3483:   Function StrEnsureNoSuffix( const Suffix, Text : AnsiString ) : AnsiString
3484:   Function StrEnsurePrefix( const Prefix, Text : AnsiString ) : AnsiString
3485:   Function StrEnsureSuffix( const Suffix, Text : AnsiString ) : AnsiString
3486:   Function StrEscapedToString( const S : AnsiString ) : AnsiString
3487:   Function JStrLower( const S : AnsiString ) : AnsiString
3488:   Procedure StrLowerInPlace( var S : Ansistring)
3489:   //Procedure StrLowerBuff( S : PAnsichar )
3490:   Procedure JStrMove( var Dest:Ansistring; const Source:Ansistring;const ToIndex,FromIndex,Count:Integer );

```

```

3491: Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar ) : AnsiString
3492: Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar ) : AnsiString
3493: Function StrProper( const S : AnsiString ) : AnsiString
3494: //Procedure StrProperBuff( S : PAnsiChar )
3495: Function StrQuote( const S : AnsiString; C : AnsiChar ) : AnsiString
3496: Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3497: Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3498: Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3499: Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar ) : AnsiString
3500: Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3501: Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3502: Function StrRepeat( const S : AnsiString; Count : Integer ) : AnsiString
3503: Function StrRepeatLength( const S : AnsiString; const L : Integer ) : AnsiString
3504: Function StrReverse( const S : AnsiString ) : AnsiString
3505: Procedure StrReverseInPlace( var S : AnsiString )
3506: Function StrSingleQuote( const S : AnsiString ) : AnsiString
3507: Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet ) : AnsiString
3508: Function StrStringToEscaped( const S : AnsiString ) : AnsiString
3509: Function StrStripNonNumberChars( const S : AnsiString ) : AnsiString
3510: Function StrToHex( const Source : AnsiString ) : AnsiString
3511: Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar ) : AnsiString
3512: Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3513: Function StrTrimCharRight( const S : AnsiString; C : AnsiChar ) : AnsiString
3514: Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3515: Function StrTrimQuotes( const S : AnsiString ) : AnsiString
3516: Function JStrUpper( const S : AnsiString ) : AnsiString
3517: Procedure StrUpperInPlace( var S : AnsiString )
3518: //Procedure StrUpperBuff( S : PAnsiChar )
3519: Function StrOemToAnsi( const S : AnsiString ) : AnsiString
3520: Function StrAnsiToOem( const S : AnsiString ) : AnsiString
3521: Procedure StrAddRef( var S : AnsiString )
3522: Function StrAllocSize( const S : AnsiString ) : Longint
3523: Procedure StrDecRef( var S : AnsiString )
3524: //Function StrLen( S : PAnsiChar ) : Integer
3525: Function StrLength( const S : AnsiString ) : Longint
3526: Function StrRefCount( const S : AnsiString ) : Longint
3527: Procedure StrResetLength( var S : AnsiString )
3528: Function StrCharCount( const S : AnsiString; C : AnsiChar ) : Integer
3529: Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet ) : Integer
3530: Function StrSubsCount( const S, Subs : AnsiString ) : Integer
3531: Function StrCompare( const S1, S2 : AnsiString ) : Integer
3532: Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer ) : Integer
3533: //Function StrFillChar( const C : AnsiChar; Count : Integer ) : AnsiString;
3534: Function StrFillChar1( const C : Char; Count : Integer ) : AnsiString;
3535: Function StrFillChar( const C : Char; Count : Integer): string ')';
3536: Function IntFillChar( const I: Integer; Count: Integer): string ')';
3537: Function ByteFillChar( const B: Byte; Count: Integer): string ')';
3538: Function ArrFillChar( const AC: Char; Count: Integer): TCharArray';
3539: Function ArrByteFillChar( const AB: Char; Count: Integer): TByteArray;
3540: Function StrFind( const Substr, S : AnsiString; const Index : Integer ) : Integer
3541: //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString ) : Boolean
3542: Function StrIndex( const S : AnsiString; const List : array of AnsiString ) : Integer
3543: Function StrILastPos( const SubStr, S : AnsiString ) : Integer
3544: Function StrIPos( const SubStr, S : AnsiString ) : Integer
3545: Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString ) : Boolean
3546: Function StrLastPos( const SubStr, S : AnsiString ) : Integer
3547: Function StrMatch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3548: Function StrMatches( const Substr, S : AnsiString; const Index : Integer ) : Boolean
3549: Function StrNIPos( const S, SubStr : AnsiString; N : Integer ) : Integer
3550: Function StrNPos( const S, SubStr : AnsiString; N : Integer ) : Integer
3551: Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString ) : Integer
3552: Function StrSearch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3553: //Function StrAfter( const SubStr, S : AnsiString ) : AnsiString
3554: //Function StrBefore( const SubStr, S : AnsiString ) : AnsiString
3555: Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar ) : AnsiString
3556: Function StrChopRight( const S : AnsiString; N : Integer ) : AnsiString
3557: Function StrLeft( const S : AnsiString; Count : Integer ) : AnsiString
3558: Function StrMid( const S : AnsiString; Start, Count : Integer ) : AnsiString
3559: Function StrRestOf( const S : AnsiString; N : Integer ) : AnsiString
3560: Function StrRight( const S : AnsiString; Count : Integer ) : AnsiString
3561: Function CharEqualNoCase( const C1, C2 : AnsiChar ) : Boolean
3562: Function CharIsAlpha( const C : AnsiChar ) : Boolean
3563: Function CharIsAlphaNum( const C : AnsiChar ) : Boolean
3564: Function CharIsBlank( const C : AnsiChar ) : Boolean
3565: Function CharIsControl( const C : AnsiChar ) : Boolean
3566: Function CharIsDelete( const C : AnsiChar ) : Boolean
3567: Function CharIsDigit( const C : AnsiChar ) : Boolean
3568: Function CharIsLower( const C : AnsiChar ) : Boolean
3569: Function CharIsNumberChar( const C : AnsiChar ) : Boolean
3570: Function CharIsPrintable( const C : AnsiChar ) : Boolean
3571: Function CharIsPunctuation( const C : AnsiChar ) : Boolean
3572: Function CharIsReturn( const C : AnsiChar ) : Boolean
3573: Function CharIsSpace( const C : AnsiChar ) : Boolean
3574: Function CharIsUpper( const C : AnsiChar ) : Boolean
3575: Function CharIsWhiteSpace( const C : AnsiChar ) : Boolean
3576: Function CharType( const C : AnsiChar ) : Word
3577: Function CharHex( const C : AnsiChar ) : Byte
3578: Function CharLower( const C : AnsiChar ) : AnsiChar
3579: Function CharUpper( const C : AnsiChar ) : AnsiChar

```

```

3580: Function CharToggleCase( const C : AnsiChar ) : AnsiChar
3581: Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer ) : Integer
3582: Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer ) : Integer
3583: Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer ) : Integer
3584: Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar ) : Integer
3585: Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean )
3586: Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean )
3587: Function StringsToStr( const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool ):AnsiString;
3588: Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean )
3589: Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean )
3590: Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean )
3591: Function AddStringToStrings( const S:AnsiString;Strings:TStrings; const Unique:Boolean ):Boolean
3592: Function BooleanToStr( B : Boolean ) : AnsiString
3593: Function FileToString( const FileName : AnsiString ) : AnsiString
3594: Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean )
3595: Function StrToken( var S : AnsiString; Separator : AnsiChar ) : AnsiString
3596: Procedure StrTokens( const S : AnsiString; const List : TStrings )
3597: Procedure StrTokenToStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings )
3598: //Function StrWord( var S : PAnsiChar; out Word : AnsiString ) : Boolean
3599: Function StrToFloatSafe( const S : AnsiString ) : Float
3600: Function StrToIntSafe( const S : AnsiString ) : Integer
3601: Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer );
3602: Function ArrayOf( List : TStrings ) : TDynStringArray;
3603: EJclStringError', 'EJclError
3604: function IsClass(Address: TObject): Boolean;
3605: function IsObject(Address: TObject): Boolean;
3606: // Console Utilities
3607: //function ReadKey: Char;
3608: function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3609: function JclGUIDToString(const GUID: TGUID): string;
3610: function JclStringToGUID(const S: string): TGUID;
3611: end;
3612:
3613:
3614: ***** uPSI_JvDBUtil;
3615: Procedure ExecuteSQLScript( Base:TDataBase; const Script: string; const Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer )
3616: Function GetQueryResult( const DatabaseName, SQL : string ) : Variant
3617: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant; const AResultName : string ) : Variant
3618: //Function StrFieldDesc( Field : FLDDesc ) : string
3619: Function Var2Type( V : Variant; const VarType : Integer ) : Variant
3620: Procedure CopyRecord( DataSet : TDataSet )
3621: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string; MasterField : Word; ModOp, DelOp : RINTqual )
3622: Procedure AddMasterPassword( Table : TTable; pswd : string )
3623: Procedure PackTable( Table : TTable )
3624: Procedure PackEncryptedTable( Table : TTable; pswd : string )
3625: Function EncodeQuotes( const S : string ) : string
3626: Function Cmp( const S1, S2 : string ) : Boolean
3627: Function SubStr( const S : string; const Index : Integer; const Separator : string ) : string
3628: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string ) : string
3629: Function ReplaceString( S : string; const OldPattern, NewPattern : string ) : string
3630: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer )
3631: *****uPSI_JvJvBDEUtils;*****
3632: //JvBDEUtils
3633: Function CreateDbLocate : TJvLocateObject
3634: //Function CheckOpen( Status : DBIResult ) : Boolean
3635: Procedure FetchAllRecords( DataSet : TBDEDataSet )
3636: Function TransActive( Database : TDatabase ) : Boolean
3637: Function AsyncQrySupported( Database : TDatabase ) : Boolean
3638: Function GetQuoteChar( Database : TDatabase ) : string
3639: Procedure ExecuteQuery( const DbName, QueryText : string )
3640: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string )
3641: Function FieldLogicMap( FldType : TFieldType ) : Integer
3642: Function FieldsSubtypeMap( FldType : TFieldType ) : Integer Value : string; Buffer : Pointer )
3643: Function GetAliasPath( const AliasName : string ) : string
3644: Function IsDirectory( const DatabaseName : string ) : Boolean
3645: Function GetBdeDirectory : string
3646: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent ) : Boolean
3647: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value,FieldName : string ) : Boolean
3648: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value,FieldName : string ) : Boolean
3649: Function DataSetRecNo( DataSet : TDataSet ) : Longint
3650: Function DataSetRecordCount( DataSet : TDataSet ) : Longint
3651: Function DataSetPositionStr( DataSet : TDataSet ) : string
3652: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean )
3653: Function CurrentRecordDeleted( DataSet : TBDEDataSet ) : Boolean
3654: Function IsFilterApplicable( DataSet : TDataSet ) : Boolean
3655: Function IsBookmarkStable( DataSet : TBDEDataSet ) : Boolean
3656: Procedure SetIndex( Table : TTable; const IndexFieldNames : string )
3657: Procedure RestoreIndex( Table : TTable )
3658: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const )
3659: Procedure PackTable( Table : TTable )
3660: Procedure ReindexTable( Table : TTable )
3661: Procedure BdeflushBuffers
3662: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer ) : Pointer
3663: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string )
3664: Procedure DbNotSupported
3665: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint )

```

```

3666: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
  AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint );
3667: Procedure
  ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3668: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3669: ****uPSI_JvDateUtil;
3670: function CurrentYear: Word;
3671: function IsLeapYear(AYear: Integer): Boolean;
3672: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3673: function FirstDayOfPrevMonth: TDateTime;
3674: function LastDayOfPrevMonth: TDateTime;
3675: function FirstDayOfNextMonth: TDateTime;
3676: function ExtractDay(ADate: TDateTime): Word;
3677: function ExtractMonth(ADate: TDateTime): Word;
3678: function ExtractYear(ADate: TDateTime): Word;
3679: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3680: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3681: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3682: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3683: function ValidDiff(ADate: TDateTime): Boolean;
3684: procedure DateDiff(Datel, Date2: TDateTime; var Days, Months, Years: Word);
3685: function MonthsBetween(Datel, Date2: TDateTime): Double;
3686: function DaysInPeriod(Datel, Date2: TDateTime): Longint;
3687: { Count days between Datel and Date2 + 1, so if Datel = Date2 result = 1 }
3688: function DaysBetween(Datel, Date2: TDateTime): Longint;
3689: { The same as previous but if Date2 < Datel result = 0 }
3690: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3691: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3692: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3693: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3694: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3695: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3696: { String to date conversions }
3697: function GetDateOrder(const DateFormat: string): TDateOrder;
3698: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3699: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3700: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3701: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3702: function DefDateFormat(FourDigitYear: Boolean): string;
3703: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3704: -----
3705: ***** JvUtils*****
3706: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3707: function GetWordOnPos(const S: string; const P: Integer): string;
3708: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3709: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3710: { SubStr returns substring from string, S, separated with Separator string}
3711: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3712: { SubStrEnd same to previous function but Index numerated from the end of string }
3713: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3714: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3715: function SubWord(P: PChar; var P2: PChar): string;
3716: { NumberByWord returns the text representation of
3717:   the number, N, in normal russian language. Was typed from Monitor magazine }
3718: function NumberByWord(const N: Longint): string;
3719: // function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3720: //the symbol Pos is pointed. Lines separated with #13 symbol }
3721: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3722: { GetXYByPos is same to previous function, but returns X position in line too}
3723: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3724: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3725: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3726: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3727: function ConcatSep(const S, S2, Separator: string): string;
3728: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3729: function ConcatLeftSep(const S, S2, Separator: string): string;
3730: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3731: function MinimizeString(const S: string; const MaxLen: Integer): string;
3732: { Next 4 function for russian chars transliterating.
3733:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3734: procedure Dos2Win(var S: string);
3735: procedure Win2Dos(var S: string);
3736: function Dos2WinRes(const S: string): string;
3737: function Win2DOSRes(const S: string): string;
3738: function Win2Koi(const S: string): string;
3739: { Spaces returns string consists on N space chars }
3740: function Spaces(const N: Integer): string;
3741: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3742: function AddSpaces(const S: string; const N: Integer): string;
3743: { function LastDate for russian users only } { returns date relative to current date: '' }
3744: function LastDate(const Dat: TDateTime): string;
3745: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3746: function CurrencyToStr(const Cur: currency): string;
3747: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3748: function Cmp(const S1, S2: string): Boolean;
3749: { StringCat add S2 string to S1 and returns this string }
3750: function StringCat(var S1: string; S2: string): string;
3751: { HasChar returns True, if Char, Ch, contains in string, S }
3752: function HasChar(const Ch: Char; const S: string): Boolean;

```

```

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

```

```

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

```

```

3926: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3927:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3928: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3929:   Buttons: TMsgDlgButtons; DefButton: TMsgDlgBtn; HelpContext: Integer; Control: TWinControl): Integer;
3930: { Delay stop program execution to MSec msec }
3931: procedure Delay(MSec: Longword);
3932: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3933: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3934: procedure EnableMenuItem(Menuitem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3935: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3936: function PanelBorder(Panel: TCustomPanel): Integer;
3937: function Pixels(Control: TControl; APixels: Integer): Integer;
3938: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3939: procedure Error(const Msg: string);
3940: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3941:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3942: {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:Blue>blue</i>'}
3943: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3944:   const HideSelColor: Boolean): string;
3945: function ItemHtWidth(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3946:   const HideSelColor: Boolean): Integer;
3947: function ItemHtPlain(const Text: string): string;
3948: { ClearList - clears list of TObject }
3949: procedure ClearList(List: TList);
3950: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3951: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3952: { RTTI support }
3953: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3954: function GetPropStr(Obj: TObject; const PropName: string): string;
3955: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3956: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3957: procedure PrepareIniSection(SS: TStrings);
3958: { following functions are not documented because they are don't work properly, so don't use them }
3959: {$IFDEF COMPILER2}
3960: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3961: {$ENDIF}
3962:
3963: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3964: begin
3965:   Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl )
3966:   Procedure BoxMoveAllItems( SrcList, DstList : TWinControl )
3967:   Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
3968:   Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer )
3969:   Procedure BoxMoveSelected( List : TWinControl; Items : TStrings )
3970:   Procedure BoxSetItem( List : TWinControl; Index : Integer )
3971:   Function BoxGetFirstSelection( List : TWinControl ) : Integer
3972:   Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer ) : Boolean
3973: end;
3974:
3975: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3976: begin
3977:   Const ('MaxInitStrNum','LongInt'( 9 );
3978:   Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
: array of AnsiString; MaxSplit : Integer ) : Integer
3979:   Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
string; MaxSplit : Integer ) : Integer
3980:   Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3981:   Function JvAnsiStrStrip( S : AnsiString ) : AnsiString
3982:   Function JvStrStrip( S : string ) : string
3983:   Function GetString( var Source : AnsiString; const Separator : AnsiString ) : AnsiString
3984:   Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar ) : AnsiString
3985:   Function BuildPathName( const PathName, FileName : AnsiString ) : AnsiString
3986:   Function StrEatWhiteSpace( const S : string ) : string
3987:   Function HexToAscii( const S : AnsiString ) : AnsiString
3988:   Function AsciiToHex( const S : AnsiString ) : AnsiString
3989:   Function StripQuotes( const S1 : AnsiString ) : AnsiString
3990:   Function ValidNumericLiteral( S1 : PAnsiChar ) : Boolean
3991:   Function ValidIntLiteral( S1 : PAnsiChar ) : Boolean
3992:   Function ValidHexLiteral( S1 : PAnsiChar ) : Boolean
3993:   Function HexPCharToInt( S1 : PAnsiChar ) : Integer
3994:   Function ValidStringLiteral( S1 : PAnsiChar ) : Boolean
3995:   Function StripCharQuotes( S1 : PAnsiChar ) : AnsiString
3996:   Function JvValidIdentifierA( S1 : PAnsiChar ) : Boolean
3997:   Function JvValidIdentifier( S1 : string ) : Boolean
3998:   Function JvEndChar( X : AnsiChar ) : Boolean
3999:   Procedure JvGetToken( S1, S2 : PAnsiChar )
4000:   Function IsExpressionKeyword( S1 : PAnsiChar ) : Boolean
4001:   Function IsKeyword( S1 : PAnsiChar ) : Boolean
4002:   Function JvValidVarReference( S1 : PAnsiChar ) : Boolean
4003:   Function GetParenthesis( S1, S2 : PAnsiChar ) : Boolean
4004:   Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar )
4005:   Procedure JvEatWhitespaceChars( S1 : PAnsiChar );
4006:   Procedure JvEatWhitespaceChars1( S1 : PWideChar );
4007:   Function GetTokenCount : Integer
4008:   Procedure ResetTokenCount
4009: end;
4010:

```

```

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

```

```

4100: { Copy2Symb returns a substring of a string S from begining to first white space. }
4101: function Copy2SpaceDel(var S: string): string;
4102: { Copy2SpaceDel returns a substring of a string S from begining to first white space and removes this substring from S. }
4103: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4104: { Returns string, with the first letter of each word in uppercase, all other letters in lowercase. Words are delimited by WordDelims. }
4105: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4106: { WordCount given a set of word delimiters, returns number of words in S. }
4107: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4108: { Given a set of word delimiters, returns start position of N'th word in S. }
4109: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4110: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TCharSet; var Pos: Integer): string;
4111: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4112: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word delimiters, return the N'th word in S. }
4113: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4114: { ExtractSubstr given set of word delimiters, returns the substring from S, that started from position Pos. }
4115: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4116: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4117: function QuotedString(const S: string; Quote: Char): string;
4118: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4119: function ExtractQuotedString(const S: string; Quote: Char): string;
4120: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string, and reduces pairs of Quote characters within the quoted string to a single character. }
4121: function FindPart(const HelpWilds, InputStr: string): Integer;
4122: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4123: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4124: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4125: function XorString(const Key, Src: ShortString): ShortString;
4126: function XorEncode(const Key, Source: string): string;
4127: function XorDecode(const Key, Source: string): string;
4128: { ** Command line routines ** }
4129: {$IFDEF COMPILER4_UP}
4130: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet; IgnoreCase: Boolean): Boolean;
4131: {$ENDIF}
4132: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4133: { ** Numeric string handling routines ** }
4134: function Numb2USA(const S: string): string;
4135: { Numb2USA converts numeric string S to USA-format. }
4136: function Dec2Hex(N: Longint; A: Byte): string;
4137: function D2H(N: Longint; A: Byte): string;
4138: { Dec2Hex converts the given value to a hexadecimal string representation with the minimum number of digits (A) specified. }
4139: function Hex2Dec(const S: string): Longint;
4140: function H2D(const S: string): Longint;
4141: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4142: function Dec2Numb(N: Longint; A, B: Byte): string;
4143: { Dec2Numb converts the given value to a string representation with the base equal to B and with the minimum number of digits (A) specified. }
4144: function Numb2Dec(S: string; B: Byte): Longint;
4145: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4146: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4147: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4148: function IntToRoman(Value: Longint): string;
4149: { IntToRoman converts the given value to a roman numeric string representation. }
4150: function RomanToInt(const S: string): Longint;
4151: { RomanToInt converts the given string to an integer value. If the string doesn't contain a valid roman numeric value, the 0 value is returned. }
4152: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4153: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4154: ***** JvFileUtil;*****
4155: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4156: procedure CopyFileEx(const FileName, DestName: string; OverwriteReadOnly, ShellDialog: Boolean; ProgressControl: TControl);
4157: procedure MoveFile(const FileName, DestName: TFileName);
4158: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4159: {$IFDEF COMPILER4_UP}
4160: function GetFileSize(const FileName: string): Int64;
4161: {$ELSE}
4162: function GetFileSize(const FileName: string): Longint;
4163: {$ENDIF}
4164: function FileDateTime(const FileName: string): TDateTime;
4165: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4166: function DeleteFiles(const FileMask: string): Boolean;
4167: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4168: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4169: function NormalDir(const DirName: string): string;
4170: function RemoveBackSlash(const DirName: string): string;
4171: function ValidFileName(const FileName: string): Boolean;
4172: function DirExists(Name: string): Boolean;
4173: procedure ForceDirectories(Dir: string);
4174: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4175: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4176: {$IFDEF COMPILER4_UP} {$ENDIF}
4177: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4178: {$ENDIF}
4179: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4180: {$IFDEF COMPILER4_UP} overload; {$ENDIF}

```

```

4188: {$IFDEF COMPILER4_UP}
4189: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4190: {$ENDIF}
4191: function GetTempDir: string;
4192: function GetWindowsDir: string;
4193: function GetSystemDir: string;
4194: function BrowseDirectory(var AFolderPath:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4195: {$IFDEF WIN32}
4196: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4197: function ShortToLongFileName(const ShortName: string): string;
4198: function ShortToLongPath(const ShortName: string): string;
4199: function LongToShortFileName(const LongName: string): string;
4200: function LongToShortPath(const LongName: string): string;
4201: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4202: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4203: {$ENDIF WIN32}
4204: {$IFNDEF COMPILER3_UP}
4205: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4206: {$ENDIF}
4207: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext ) : TJvCalculatorForm;
4208: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode ) : TWinControl;
4209: Function CreatePopupCalculator( AOwner : TComponent ) : TWinControl;
4210: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean );
4211: 
4212: //*****procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4213: Procedure VariantClear( var V : Variant );
4214: Procedure VariantArrayRedim( var V : Variant; High : Integer );
4215: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer );
4216: Procedure VariantCpy( const src : Variant; var dst : Variant );
4217: Procedure VariantAdd( const src : Variant; var dst : Variant );
4218: Procedure VariantSub( const src : Variant; var dst : Variant );
4219: Procedure VariantMul( const src : Variant; var dst : Variant );
4220: Procedure VariantDiv( const src : Variant; var dst : Variant );
4221: Procedure VariantMod( const src : Variant; var dst : Variant );
4222: Procedure VariantAnd( const src : Variant; var dst : Variant );
4223: Procedure VariantOr( const src : Variant; var dst : Variant );
4224: Procedure VariantXor( const src : Variant; var dst : Variant );
4225: Procedure VariantShl( const src : Variant; var dst : Variant );
4226: Procedure VariantShr( const src : Variant; var dst : Variant );
4227: Function VariantAdd2( const V1 : Variant; const V2 : Variant ) : Variant;
4228: Function VariantSub2( const V1 : Variant; const V2 : Variant ) : Variant;
4229: Function VariantMul2( const V1 : Variant; const V2 : Variant ) : Variant;
4230: Function VariantDiv2( const V1 : Variant; const V2 : Variant ) : Variant;
4231: Function VariantMod2( const V1 : Variant; const V2 : Variant ) : Variant;
4232: Function VariantAnd2( const V1 : Variant; const V2 : Variant ) : Variant;
4233: Function VariantOr2( const V1 : Variant; const V2 : Variant ) : Variant;
4234: Function VariantXor2( const V1 : Variant; const V2 : Variant ) : Variant;
4235: Function VariantShl2( const V1 : Variant; const V2 : Variant ) : Variant;
4236: Function VariantShr2( const V1 : Variant; const V2 : Variant ) : Variant;
4237: Function VariantNot( const V1 : Variant ) : Variant;
4238: Function VariantNeg( const V1 : Variant ) : Variant;
4239: Function VariantGetElement( const V : Variant; il : integer ) : Variant;
4240: Function VariantGetElement1( const V : Variant; il, i2 : integer ) : Variant;
4241: Function VariantGetElement2( const V : Variant; il, i2, i3 : integer ) : Variant;
4242: Function VariantGetElement3( const V : Variant; il, i2, i3, i4 : integer ) : Variant;
4243: Function VariantGetElement4( const V : Variant; il, i2, i3, i4, i5 : integer ) : Variant;
4244: Procedure VariantPutElement( var V : Variant; const data : Variant; il : integer );
4245: Procedure VariantPutElement1( var V : Variant; const data : Variant; il, i2 : integer );
4246: Procedure VariantPutElement2( var V : Variant; const data : Variant; il, i2, i3 : integer );
4247: Procedure VariantPutElement3( var V : Variant; const data : Variant; il, i2, i3, i4 : integer );
4248: Procedure VariantPutElement4( var V : Variant; const data : Variant; il, i2, i3, i4, i5 : integer );
4249: end;
4250: 
4251: *****unit uPSI_JvgUtils;*****
4252: function IsEven(I: Integer): Boolean;
4253: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4254: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4255: procedure SwapInt(var I1, I2: Integer);
4256: function Spaces(Count: Integer): string;
4257: function DupStr(const Str: string; Count: Integer): string;
4258: function DupChar(C: Char; Count: Integer): string;
4259: procedure Msg(const AMsg: string);
4260: function RectW(R: TRect): Integer;
4261: function RectH(R: TRect): Integer;
4262: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4263: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4264: function IsItFilledBitmap(Bmp: TBitmap): Boolean;
4265: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4266:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4267: procedure DrawTextInRect(DC: HDC; R: TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4268: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4269:   Style: TglTextStyle; ADelinedate, ASupress3D: Boolean; FontColor, DelinColor, HighlightColor, ShadowColor: TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4270: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4271: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4272:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint; ATransparent: Boolean): TRect;
4273: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4274: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4275: procedure DrawBitmapExt(DC: HDC; { DC - background & result }

```

```

4276:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4277:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4278:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4279:   procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4280:     SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4281:     BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4282:     ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4283:   procedure BringParentWindowToTop(Wnd: TWinControl);
4284:   function GetParentForm(Control: TControl): TForm;
4285:   procedure GetWindowImageFrom(Control: TWinControl; X, Y: Integer; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4286:   procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4287:   procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4288:   function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4289:   function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4290:   procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4291:   function CalcMathString(AExpression: string): Single;
4292:   function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4293:   function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4294:   function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4295:   procedure TypeStringOnKeyboard(const S: string);
4296:   function NextStringGridCell(Grid: TStringGrid): Boolean;
4297:   procedure DrawTextExtAligned(Canvas: TCanvas; const
4298:     Text: string; R: TRect; Alignment: TglAlignment; WordWrap: Boolean);
4299:   procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4300:   procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4301:   function ComponentToString(Component: TComponent): string;
4302:   procedure StringToComponent(Component: TComponent; const Value: string);
4303:   function PlayWaveResource(const ResName: string): Boolean;
4304:   function UserName: string;
4305:   function ComputerName: string;
4306:   function ExpandString(const Str: string; Len: Integer): string;
4307:   function Transliterate(const Str: string; RusToLat: Boolean): string;
4308:   function IsSmallFonts: Boolean;
4309:   function SystemColorDepth: Integer;
4310:   function GetFileTypeJ(const FileName: string): TglFileType;
4311:   Function GetFileType(hfile : THandle) : DWORD';
4312:   function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4313:   function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4314:
4315: { **** Utility routines of unit classes }
4316:   function LineStart(Buffer, BufPos: PChar): PChar;
4317:   function ExtractStrings(Separators,WhiteSpace: TSysCharSet; Content: PChar; '+
4318:   'Strings: TStrings): Integer;
4319:   Function TestStreamFormat(Stream : TStream) : TStreamOriginalFormat;
4320:   Procedure RegisterClass(AClass : TPersistentClass);
4321:   Procedure RegisterClasses(AClasses : array of TPersistentClass);
4322:   Procedure RegisterClassAlias(AClass : TPersistentClass; const Alias: string);
4323:   Procedure UnRegisterClass(AClass : TPersistentClass);
4324:   Procedure UnRegisterClasses(AClasses : array of TPersistentClass);
4325:   Procedure UnRegisterModuleClasses(Module : HMODULE);
4326:   Function FindGlobalComponent(const Name: string): TComponent;
4327:   Function IsUniqueGlobalComponentName(const Name: string): Boolean;
4328:   Function InitInheritedComponent(Instance: TComponent; RootAncestor: TClass): Boolean;
4329:   Function InitComponentRes(const ResName: string; Instance: TComponent): Boolean;
4330:   Function ReadComponentRes(const ResName: string; Instance: TComponent): TComponent;
4331:   Function ReadComponentResEx(HInstance: THandle; const ResName: string): TComponent;
4332:   Function ReadComponentResFile(const FileName: string; Instance: TComponent): TComponent;
4333:   Procedure WriteComponentResFile(const FileName: string; Instance: TComponent);
4334:   Procedure GlobalFixupReferences;
4335:   Procedure GetFixupReferenceNames(Root: TComponent; Names: TStrings);
4336:   Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName: string; Names: TStrings);
4337:   Procedure RedirectFixupReferences(Root: TComponent; const OldRootName, NewRootName: string);
4338:   Procedure RemoveFixupReferences(Root: TComponent; const RootName: string);
4339:   Procedure RemoveFixups(Instance: TPersistent);
4340:   Function FindNestedComponent(Root: TComponent; const NamePath: string): TComponent;
4341:   Procedure BeginGlobalLoading;
4342:   Procedure NotifyGlobalLoading;
4343:   Procedure EndGlobalLoading;
4344:   Function GetUltimateOwner1(ACollection: TCollection): TPersistent;
4345:   Function GetUltimateOwner(APersistent: TPersistent): TPersistent;
4346:   // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4347:   //Function MakeObjectInstance( Method : TWndMethod ) : Pointer
4348:   Procedure FreeObjectInstance(ObjectInstance: Pointer);
4349:   // Function AllocateHWnd( Method : TWndMethod ) : HWND
4350:   Procedure DeallocateHWnd(Wnd: HWND);
4351:   Function AncestorIsValid(Ancestor: TPersistent; Root, RootAncestor: TComponent): Boolean;
4352: { **** unit uPSI_SqlTimSt and DB; ****}
4353:   Procedure VarSQLTimeStampCreate4(var aDest: Variant; const ASQLTimeStamp: TSQLTimeStamp);
4354:   Function VarSQLTimeStampCreate3: Variant;
4355:   Function VarSQLTimeStampCreate2(const AValue: string): Variant;
4356:   Function VarSQLTimeStampCreate1(const AValue: TDateTime): Variant;
4357:   Function VarSQLTimeStampCreate(const ASQLTimeStamp: TSQLTimeStamp): Variant;
4358:   Function VarSQLTimeStamp: TVarType;
4359:   Function VarIsSQLTimeStamp(const aValue: Variant): Boolean;
4360:   Function LocalToUTC(var TZInfo: TTimeZone; var Value: TSQLTimeStamp): TSQLTimeStamp //beta
4361:   Function UTCToLocal(var TZInfo: TTimeZone; var Value: TSQLTimeStamp): TSQLTimeStamp //beta
4362:   Function VarToSQLTimeStamp(const aValue: Variant): TSQLTimeStamp;
4363:   Function SQLTimeStampToStr(const Format: string; DateTime: TSQLTimeStamp): string;

```

```

4364: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp ) : integer
4365: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime ) : TSQLTimeStamp
4366: Function SQLTimeStampToDate( const DateTime : TSQLTimeStamp ) : TDateTime
4367: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp ) : Boolean
4368: Function StrToSQLTimeStamp( const S : string ) : TSQLTimeStamp
4369: Procedure CheckSQLTimeStamp( const ASQLTimeStamp : TSQLTimeStamp )
4370: Function ExtractFieldName( const Fields : string; var Pos : Integer ) : string;
4371: Function ExtractFieldName( const Fields : WideString; var Pos : Integer ) : WideString;
4372: //Procedure RegisterFields( const FieldClasses : array of TFieldClass )
4373: Procedure DatabaseError( const Message : WideString; Component : TComponent )
4374: Procedure DatabaseErrorFmt( const Message:WideString; const Args:array of const;Component:TComponent )
4375: Procedure DisposeMem( var Buffer, Size : Integer )
4376: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer ) : Boolean
4377: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4378: Function VarTypeToDataType( VarType : Integer ) : TFieldType
4379: *****unit JvStrings;*****
4380: {template functions}
4381: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4382: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4383: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4384: function RemoveMasterBlocks(const SourceStr: string): string;
4385: function RemoveFields(const SourceStr: string): string;
4386: {http functions}
4387: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4388: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4389: {set functions}
4390: procedure SplitSet(AText: string; AList: TStringList);
4391: function JoinSet(AList: TStringList): string;
4392: function FirstOfSet(const AText: string): string;
4393: function LastOfSet(const AText: string): string;
4394: function CountOfSet(const AText: string): Integer;
4395: function SetRotateRight(const AText: string): string;
4396: function SetRotateLeft(const AText: string): string;
4397: function SetPick(const AText: string; AIndex: Integer): string;
4398: function SetSort(const AText: string): string;
4399: function SetUnion(const Set1, Set2: string): string;
4400: function SetIntersect(const Set1, Set2: string): string;
4401: function SetExclude(const Set1, Set2: string): string;
4402: {replace any <,> etc by &lt;,&gt;}
4403: function XMLSafe(const AText: string): string;
4404: {simple hash, Result can be used in Encrypt}
4405: function Hash(const AText: string): Integer;
4406: { Base64 encode and decode a string }
4407: function B64Encode(const S: AnsiString): AnsiString;
4408: function B64Decode(const S: AnsiString): AnsiString;
4409: {Basic encryption from a Borland Example}
4410: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4411: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4412: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4413: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4414: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4415: procedure CSVToTags(Src, Dst: TStringList);
4416: // converts a csv list to a tagged string list
4417: procedure TagsToCSV(Src, Dst: TStringList);
4418: // converts a tagged string list to a csv list
4419: // only fieldnames from the first record are scanned in the other records
4420: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4421: {selects akey=avalue from Src and returns recordset in Dst}
4422: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4423: {filters Src for akey=avalue}
4424: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4425: {orders a tagged Src list by akey}
4426: function PosStr(const FindString, SourceString: string);
4427: StartPos: Integer = 1): Integer;
4428: { PosStr searches the first occurrence of a substring FindString in a string
4429: given by SourceString with case sensitivity (upper and lower case characters
4430: are differed). This function returns the index value of the first character
4431: of a specified substring from which it occurs in a given string starting with
4432: StartPos character index. If a specified substring is not found Q_PosStr
4433: returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4434: function PosStrLast(const FindString, SourceString: string): Integer;
4435: {finds the last occurrence}
4436: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4437: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4438: { PosText searches the first occurrence of a substring FindString in a string
4439: given by SourceString without case sensitivity (upper and lower case characters are not differed). This
4440: function returns the index value of the first character of a specified substring from which it occurs in a
4441: given string starting with Start
4442: function PosTextLast(const FindString, SourceString: string): Integer;
4443: {finds the last occurrence}
4444: function NameValuesToXML(const AText: string): string;
4445: {$IFDEF MSWINDOWS}
4446: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4447: {$ENDIF MSWINDOWS}
4448: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4449: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4450: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4451: procedure SaveString(const AFile, AText: string);
4452: Procedure SaveStringasFile( const AFile, AText : string)

```

```

4451: function LoadStringJ(const AFile: string): string;
4452: Function LoadStringOfFile( const AFile : string ) : string
4453: Procedure SaveStringToFile( const AFile, AText : string)
4454: Function LoadStringFromFile( const AFile : string ) : string
4455: function HexToColor(const AText: string): TColor;
4456: function UppercaseHTMLTags(const AText: string): string;
4457: function LowercaseHTMLTags(const AText: string): string;
4458: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4459: function RelativePath(const ASrc, ADst: string): string;
4460: function GetToken(var Start: Integer; const SourceText: string): string;
4461: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4462: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4463: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4464: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4465: // parses the beginning of an attribute: space + alpha character
4466: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4467: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4468: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4469: // parses all name=value attributes to the attributes TStringList
4470: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4471: // checks if a name="value" pair exists and returns any value
4472: function GetStrValue(const AText, AName, ADefault: string): string;
4473: // retrieves string value from a line like:
4474: // name="jan verhoeven" email="janl dott verhoeven att wxs dott nl"
4475: // returns ADefault when not found
4476: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4477: // same for a color
4478: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4479: // same for an Integer
4480: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4481: // same for a float
4482: function GetBoolValue(const AText, AName: string): Boolean;
4483: // same for Boolean but without default
4484: function GetValue(const AText, AName: string): string;
4485: //retrieves string value from a line like: name="jan verhoeven" email="janl verhoeven att wxs dott nl"
4486: procedure SetValue(var AText: string; const AName, AValue: string);
4487: // sets a string value in a line
4488: procedure DeleteValue(var AText: string; const AName: string);
4489: // deletes a AName="value" pair from AText
4490: procedure GetNames(AText: string; AList: TStringList);
4491: // get a list of names from a string with name="value" pairs
4492: function GetHTMLColor(AColor: TColor): string;
4493: // converts a color value to the HTML hex value
4494: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4495: // finds a string backward case sensitive
4496: function BackPostText(Start: Integer; const FindString, SourceString: string): Integer;
4497: // finds a string backward case insensitive
4498: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string);
4499: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4500: // finds a text range, e.g. <TD>....</TD> case sensitive
4501: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string);
4502: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4503: // finds a text range, e.g. <TD>....</td> case insensitive
4504: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string);
4505: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4506: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4507: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string);
4508: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4509: // finds a text range backward, e.g. <TD>....</td> case insensitive
4510: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4511: var RangeEnd: Integer): Boolean;
4512: // finds a HTML or XML tag: <....>
4513: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string);
4514: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4515: // finds the innerText between opening and closing tags
4516: function Easter(NYear: Integer): TDateTime;
4517: // returns the easter date of a year.
4518: function GetWeekNumber(Today: TDateTime): string;
4519: //gets a datecode. Returns year and weeknumber in format: YYWW
4520: function ParseNumber(const S: string): Integer;
4521: // parse number returns the last position, starting from 1
4522: function ParseDate(const S: string): Integer;
4523: // parse a SQL style date string from positions 1,
4524: // starts and ends with #
4525:
4526: *****unit JvJCLUtils;*****
4527:
4528: function VarIsInt(Value: Variant): Boolean;
4529: // VarIsInt returns VarIsOrdinal-[varBoolean]
4530: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4531: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4532: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4533: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4534: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4535: function GetWordOnPos(const S: string; const P: Integer): string;
4536: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4537: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4538: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4539: { GetWordOnPosEx working like GetWordOnPos function, but

```

```

4540: also returns Word position in iBeg, iEnd variables }
4541: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4542: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4543: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4544: function GetNextWordPosExW(const Text: WideString; StartIndex: Integer; var iBeg, iEnd: Integer): WideString;
4545: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4546: { GetEndPosCaret returns the caret position of the last char. For the position
4547: after the last char of Text you must add 1 to the returned X value. }
4548: procedure GetEndPosCaretW(const Text: WideString; CaretX, CaretY: Integer; var X, Y: Integer);
4549: { GetEndPosCaret returns the caret position of the last char. For the position
4550: after the last char of Text you must add 1 to the returned X value. }
4551: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4552: function SubStrBySeparator(const S:string;const Index:Integer;const
Separator:string;startIndex:Int=1):string;
4553: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
Separator:WideString;startIndex:Int:WideString;
4554: { SubStrEnd same to previous function but Index numerated from the end of string }
4555: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4556: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4557: function SubWord(P: PChar; var P2: PChar): string;
4558: function CurrencyByWord(Value: Currency): string;
4559: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4560: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4561: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4562: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4563: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4564: { ReplaceString searches for all substrings, OldPattern,
4565: in a string, S, and replaces them with NewPattern }
4566: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4567: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
WideString;StartIndex:Integer=1):WideString;
4568: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4569: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
SUPPORTS_INLINE}
4570: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
4571: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
SUPPORTS_INLINE}
4572:
4573: { Next 4 function for russian chars transliterating.
4574: This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4575: procedure Dos2Win(var S: AnsiString);
4576: procedure Win2Dos(var S: AnsiString);
4577: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4578: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4579: function Win2Koi(const S: AnsiString): AnsiString;
4580: { FillString fills the string Buffer with Count Chars }
4581: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4582: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4583: { MoveString copies Count Chars from Source to Dest }
4584: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
inline; {$ENDIF SUPPORTS_INLINE} overload;
4585: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4586: DstStartIdx: Integer; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4587: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4588: procedure FillWideChar(var Buffer: string; Count: Integer; const Value: WideChar);
4589: { MoveWideChar copies Count WideChars from Source to Dest }
4590: procedure MoveWideChar(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
SUPPORTS_INLINE}
4591: { FillNativeChar fills Buffer with Count NativeChars }
4592: procedure FillNativeChar(var Buffer: string; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4593: { MoveWideChar copies Count WideChars from Source to Dest }
4594: procedure MoveNativeChar(const Source: string; var Dest: string; Count: Integer); // D2009 internal error {$IFDEF
SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4595: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4596: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4597: { Spaces returns string consists on N space chars }
4598: function Spaces(const N: Integer): string;
4599: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4600: function AddSpaces(const S: string; const N: Integer): string;
4601: function SpacesW(const N: Integer): WideString;
4602: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4603: { function LastDateRUS for russian users only }
4604: { returns date relative to current date: 'åââ åïý íâçàâ' }
4605: function LastDateRUS(const Dat: TDateTime): string;
4606: { CurrencyToStr format Currency, Cur, using ffcurrency float format}
4607: function CurrencyToStr(const Cur: Currency): string;
4608: { HasChar returns True, if Char, Ch, contains in string, S }
4609: function HasChar(const Ch: Char; const S: string): Boolean;
4610: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$IFDEF SUPPORTS_INLINE}
4611: function HasAnyChar(const Chars: string; const S: string): Boolean;
4612: {$IFNDEF COMPILER12_UP}
4613: function CharInSet(const Ch: AnsiChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4614: {$ENDIF ~COMPILER12_UP}
4615: function CharInSetW(const Ch: WideChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF
SUPPORTS_INLINE}
4616: function CountOfChar(const Ch: Char; const S: string): Integer;
4617: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
SUPPORTS_INLINE}

```

```

4618: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4619: function StrLICompW(S1, S2: PWideChar; MaxLen: Integer): Integer;
4620: function StrPosW(S, SubStr: PWideChar): PWideChar;
4621: function StrLenW(S: PWideChar): Integer;
4622: function TrimW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4623: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4624: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4625: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4626: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4627: Function GetBitmapPixelFormat( Bitmap : TBitmap ) : TPixelFormat
4628: Function GetPaletteBitmapFormat( Bitmap : TBitmap ) : TPixelFormat
4629: Procedure SetBitmapPixelFormat( Bitmap : TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4630: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4631: Procedure GrayscaleBitmap( Bitmap : TBitmap )
4632: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer ) : TStream
4633: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer )
4634: Function ScreenPixelFormat : TPixelFormat
4635: Function ScreenColorCount : Integer
4636: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic )
4637: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean ) : TPoint
4638: // SIRegister_TJvGradient(CL);
4639:
4640: {***** files routines}
4641: procedure SetDelimitedText(List: TStringList; const Text: string; Delimiter: Char);
4642: const
4643: {$IFDEF MSWINDOWS}
4644: DefaultCaseSensitivity = False;
4645: {$ENDIF MSWINDOWS}
4646: {$IFDEF UNIX}
4647: DefaultCaseSensitivity = True;
4648: {$ENDIF UNIX}
4649: { GenTempFileName returns temporary file name on
4650:   drive, there FileName is placed }
4651: function GenTempFileName(FileName: string): string;
4652: { GenTempFileNameExt same to previous function, but
4653:   returning filename has given extension, FileExt }
4654: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4655: { ClearDir clears folder Dir }
4656: function ClearDir(const Dir: string): Boolean;
4657: { DeleteDir clears and than delete folder Dir }
4658: function DeleteDir(const Dir: string): Boolean;
4659: { FileEqMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4660: function FileEqMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4661: { FileEqMasks returns True if file, FileName, is compatible with given Masks.
4662:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4663: function FileEqMasks(FileName, Masks: TFileName;
4664:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4665: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4666: {$IFDEF MSWINDOWS}
4667: { LZFileExpand expand file, FileSource,
4668:   into FileDest. Given file must be compressed, using MS Compress program }
4669: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4670: {$ENDIF MSWINDOWS}
4671: { FileGetInfo finds SearchRec record for specified file attributes}
4672: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4673: { HasSubFolder returns True, if folder APath contains other folders }
4674: function HasSubFolder(APath: TFileName): Boolean;
4675: { IsEmptyFolder returns True, if there are no files or
4676:   folders in given folder, APath}
4677: function IsEmptyFolder(APath: TFileName): Boolean;
4678: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4679: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4680: { AddPath returns FileName with Path, if FileName not contain any path }
4681: function AddPath(const FileName, Path: TFileName): TFileName;
4682: function AddPaths(const PathList, Path: string): string;
4683: function ParentPath(const Path: TFileName): TFileName;
4684: function FindInPath(const FileName, PathList: string): TFileName;
4685: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4686: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4687: { HasParam returns True, if program running with specified parameter, Param }
4688: function HasParam(const Param: string): Boolean;
4689: function HasSwitch(const Param: string): Boolean;
4690: function Switch(const Param: string): string;
4691: { ExePath returns ExtractFilePath(ParamStr(0)) }
4692: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4693: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4694: //function FileTimeToDate(FT: TFileTime): TDate;
4695: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4696: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4697: {*** Graphic routines }
4698: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4699: function IsTTFontSelected(const DC: HDC): Boolean;
4700: function KeyPressed(VK: Integer): Boolean;
4701: Function isKeypressed: boolean; //true if key on memo2 (shell output) is pressed
4702: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4703: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4704: {*** Color routines }
4705: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);

```

```

4706: function RGBToBGR(Value: Cardinal): Cardinal;
4707: //function ColorToPrettyName(Value: TColor): string;
4708: //function PrettyNameToColor(const Value: string): TColor;
4709: {**** other routines }
4710: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4711: function IntPower(Base, Exponent: Integer): Integer;
4712: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4713: function StrToBool(const S: string): Boolean;
4714: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4715: function VarToInt(V: Variant): Integer;
4716: function VarToFloat(V: Variant): Double;
4717: { following functions are not documented because they not work properly sometimes, so do not use them }
4718: // (rom) ReplaceStrings1, GetSubStr removed
4719: function GetLongFileName(const FileName: string): string;
4720: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4721: function GetParameter: string;
4722: function GetComputerID: string;
4723: function GetComputerName: string;
4724: {**** string routines }
4725: { ReplaceAllStrings searches for all substrings, Words,
4726:   in a string, S, and replaces them with Frases with the same Index. }
4727: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4728: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4729:   in the list, Words, and if founds, replaces this Word with string from another list,Frases, with the
4730:   same Index, and then update NewSelStart variable }
4731: function ReplaceStrings(const S:string;PosBeg:Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4732: { CountOfLines calculates the lines count in a string, S,
4733:   each line must be separated from another with Crlf sequence }
4733: function CountOfLines(const S: string): Integer;
4734: { DeleteLines deletes all lines from strings which in the words, words.
4735:   The word of will be deleted from strings. }
4736: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4737: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4738:   Lines contained only spaces also deletes. }
4739: procedure DeleteEmptyLines(Ss: TStrings);
4740: { SQLAddWhere addes or modifies existing where-statement, where,
4741:   to the strings, SQL. Note: If strings SQL alreadly contains where-statement,
4742:   it must be started on the begining of any line }
4743: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4744: {**** files routines - }
4745: {$IFDEF MSWINDOWS}
4746: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4747:   Resource can be compressed using MS Compress program}
4748: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4749: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string): Boolean;
4750: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4751: {$ENDIF MSWINDOWS}
4752: { IniReadSection read section, Section, from ini-file,
4753:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4754:   Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
4755: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4756: { LoadTextFile load text file, FileName, into string }
4757: function LoadTextFile(const FileName: TFileName): string;
4758: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4759: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4760: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4761: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4762: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4763: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4764: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4765: function RATextOutEx(Canvas: TCanvas; const R, RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4766: { CalcHeight calculate needed height for
4767:   correct output, using RATextOut or RATextOutEx functions }
4768: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4769: { Cinema draws some visual effect }
4770: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4771: { Roughed fills rect with special 3D pattern }
4772: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4773: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
4774:   and height, AWidth, AHeight and placed on a specified Index, Index in the source bitmap }
4774: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4775: { TextWidth calculate text with for writing using standard desktop font }
4776: function TextWidth(const AStr: string): Integer;
4777: { TextHeight calculate text height for writing using standard desktop font }
4778: function TextHeight(const AStr: string): Integer;
4779: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4780: procedure Error(const Msg: string);
4781: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4782:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4783: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4784: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4785:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4786: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4787:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4788: function ItemHtPlain(const Text: string): string;
4789: { ClearList - clears list of TObject }
4790: procedure ClearList(List: TList);
4791: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);

```

```

4792: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4793: { RTTI support }
4794: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4795: function GetPropStr(Obj: TObject; const PropName: string): string;
4796: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4797: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4798: procedure PrepareIniSection(Ss: TStrings);
4799: { following functions are not documented because they are don't work properly, so don't use them }
4800: // (rom) from JvBandWindows to make it obsolete
4801: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4802: // (rom) from JvBandUtils to make it obsolete
4803: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4804: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4805: function CreateIconFromClipboard: TIcon;
4806: { begin JvIconClipboardUtils } { Icon clipboard routines }
4807: function CF_ICON: Word;
4808: procedure AssignClipboardIcon(Icon: TIcon);
4809: { Real-size icons support routines (32-bit only) }
4810: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4811: function CreateRealSizeIcon(Icon: TIcon): HICON;
4812: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4813: {end JvIconClipboardUtils }
4814: function CreateScreenCompatibleDC: HDC;
4815: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4816: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4817: { begin JvRLE } // (rom) changed API for inclusion in JCL
4818: procedure RleCompressTo(InStream, OutStream: TStream);
4819: procedure RleDecompressTo(InStream, OutStream: TStream);
4820: procedure RleCompress(Stream: TStream);
4821: procedure RleDecompress(Stream: TStream);
4822: { end JvRLE } { begin JvDateUtil }
4823: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4824: function IsLeapYear(AYear: Integer): Boolean;
4825: function DaysInAMonth(const AYear, AMonth: Word): Word;
4826: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4827: function FirstDayOfPrevMonth: TDateTime;
4828: function LastDayOfPrevMonth: TDateTime;
4829: function FirstDayOfNextMonth: TDateTime;
4830: function ExtractDay(ADate: TDateTime): Word;
4831: function ExtractMonth(ADate: TDateTime): Word;
4832: function ExtractYear(ADate: TDateTime): Word;
4833: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4834: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4835: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4836: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4837: function Validate(ADate: TDateTime): Boolean;
4838: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4839: function MonthsBetween(Date1, Date2: TDateTime): Double;
4840: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4841: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4842: function DaysBetween(Date1, Date2: TDateTime): Longint;
4843: { The same as previous but if Date2 < Date1 result = 0 }
4844: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4845: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4846: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4847: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4848: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4849: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
4850: { String to date conversions }
4851: function GetDateOrder(const DateFormat: string): TDateOrder;
4852: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4853: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4854: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4855: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4856: //function DefDateFormat(AFourDigitYear: Boolean): string;
4857: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4858: function FormatLongDate(Value: TDateTime): string;
4859: function FormatLongDateTime(Value: TDateTime): string;
4860: { end JvDateUtil }
4861: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4862: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4863: { begin JvStrUtils } { ** Common string handling routines ** }
4864: {$IFDEF UNIX}
4865: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4866: { const ToCode, FromCode: AnsiString}): Boolean;
4867: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4868: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4869: function OemStrToAcsi(const S: AnsiString): AnsiString;
4870: function AnsiStrToOem(const S: AnsiString): AnsiString;
4871: {$ENDIF UNIX}
4872: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4873: { StrToOem translates a string from the Windows character set into the OEM character set. }
4874: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4875: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4876: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4877: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4878: function ReplaceStr(const S, Srch, Replace: string): string;
4879: { Returns string with every occurrence of Srch string replaced with Replace string. }

```

```

4880: function DelSpace(const S: string): string;
4881: { DelSpace return a string with all white spaces removed. }
4882: function DelChars(const S: string; Chr: Char): string;
4883: { DelChars return a string with all Chr characters removed. }
4884: function DelBSpace(const S: string): string;
4885: { DelBSpace trims leading spaces from the given string. }
4886: function DelESpace(const S: string): string;
4887: { DelESpace trims trailing spaces from the given string. }
4888: function DelRSpace(const S: string): string;
4889: { DelRSpace trims leading and trailing spaces from the given string. }
4890: function DelSpace1(const S: string): string;
4891: { DelSpace1 return a string with all non-single white spaces removed. }
4892: function Tab2Space(const S: string; Numb: Byte): string;
4893: { Tab2Space converts any tabulation character in the given string to the
4894:   Numb spaces characters. }
4895: function NPos(const C: string; S: string; N: Integer): Integer;
4896: { NPos searches for a N-th position of substring C in a given string. }
4897: function MakeStr(C: Char; N: Integer): string; overload;
4898: {$IFNDEF COMPILER12_UP}
4899: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4900: {$ENDIF !COMPILER12_UP}
4901: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4902: { MakeStr return a string of length N filled with character C. }
4903: function AddChar(C: Char; const S: string; N: Integer): string;
4904: { AddChar return a string left-padded to length N with characters c. }
4905: function AddCharR(C: Char; const S: string; N: Integer): string;
4906: { AddCharR return a string right-padded to length N with characters c. }
4907: function LeftStr(const S: string; N: Integer): string;
4908: { LeftStr return a string right-padded to length N with blanks. }
4909: function RightStr(const S: string; N: Integer): string;
4910: { RightStr return a string left-padded to length N with blanks. }
4911: function CenterStr(const S: string; Len: Integer): string;
4912: { CenterStr centers the characters in the string based upon the Len specified. }
4913: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4914: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4915:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4916: function CompText(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4917: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4918: function Copy2Symb(const S: string; Symb: Char): string;
4919: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4920: function Copy2SymbDel(var S: string; Symb: Char): string;
4921: { Copy2SymbDel returns a substring of a string S from beginning to first
4922:   character Symb and removes this substring from S. }
4923: function Copy2Space(const S: string): string;
4924: { Copy2Space returns a substring of a string S from beginning to first white space. }
4925: function Copy2SpaceDel(var S: string): string;
4926: { Copy2SpaceDel returns a substring of a string S from beginning to first
4927:   white space and removes this substring from S. }
4928: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4929: { Returns string, with the first letter of each word in uppercase,
4930:   all other letters in lowercase. Words are delimited by WordDelims. }
4931: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4932: { WordCount given a set of word delimiters, returns number of words in S. }
4933: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4934: { Given a set of word delimiters, returns start position of N'th word in S. }
4935: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4936: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TSysCharSet; var Pos: Integer): string;
4937: function ExtractDelimited(N: Integer; const S: string; const Delims: TSysCharSet): string;
4938: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4939:   delimiters, return the N'th word in S. }
4940: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TSysCharSet): string;
4941: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4942:   that started from position Pos. }
4943: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4944: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4945: function QuotedString(const S: string; Quote: Char): string;
4946: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4947: function ExtractQuotedString(const S: string; Quote: Char): string;
4948: { ExtractQuotedString removes the Quote characters from the beginning and
4949:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character. }
4950: function FindPart(const HelpWilds, InputStr: string): Integer;
4951: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4952: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4953: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4954: function XorString(const Key, Src: ShortString): ShortString;
4955: function XorEncode(const Key, Source: string): string;
4956: function XorDecode(const Key, Source: string): string;
4957: { ** Command line routines ** }
4958: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4959: { ** Numeric string handling routines ** }
4960: function Numb2USA(const S: string): string;
4961: { Numb2USA converts numeric string S to USA-format. }
4962: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4963: { Dec2Hex converts the given value to a hexadecimal string representation
4964:   with the minimum number of digits (A) specified. }
4965: function Hex2Dec(const S: string): Longint;
4966: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4967: function Dec2Numb(N: Int64; A, B: Byte): string;
4968: { Dec2Numb converts the given value to a string representation with the

```

```

4969:   base equal to B and with the minimum number of digits (A) specified. }
4970: function Numb2Dec(S: string; B: Byte): Int64;
4971: { Numb2Dec converts the given B-based numeric string to the corresponding
4972:   integer value. }
4973: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4974: { IntToBin converts the given value to a binary string representation
4975:   with the minimum number of digits specified. }
4976: function IntToRoman(Value: Longint): string;
4977: { IntToRoman converts the given value to a roman numeric string representation. }
4978: function RomanToInt(const S: string): Longint;
4979: { RomanToInt converts the given string to an integer value. If the string
4980:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4981: function FindNotBlankCharPos(const S: string): Integer;
4982: function FindNotBlankCharPosW(const S: WideString): Integer;
4983: function AnsiChangeCase(const S: string): string;
4984: function WideChangeCase(const S: string): string;
4985: function StartsText(const SubStr, S: string): Boolean;
4986: function EndsText(const SubStr, S: string): Boolean;
4987: function DequotedStr(const S: string; QuoteChar: Char = ''): string;
4988: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4989: {end JvStrUtils}
4990: {$IFDEF UNIX}
4991: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4992: {$ENDIF UNIX}
4993: { begin JvFileUtil }
4994: function FileDateTime(const FileName: string): TDateTime;
4995: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4996: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4997: function NormalDir(const DirName: string): string;
4998: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4999: function ValidFileName(const FileName: string): Boolean;
5000: {$IFDEF MSWINDOWS}
5001: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5002: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5003: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5004: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5005: {$ENDIF MSWINDOWS}
5006: function GetWindowsDir: string;
5007: function GetSystemDir: string;
5008: function ShortToLongFileName(const ShortName: string): string;
5009: function LongToShortFileName(const LongName: string): string;
5010: function ShortToLongPath(const ShortName: string): string;
5011: function LongToShortPath(const LongName: string): string;
5012: {$IFDEF MSWINDOWS}
5013: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
5014: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
5015: {$ENDIF MSWINDOWS}
5016: { end JvFileUtil }
5017: // Works like PtInRect but includes all edges in comparision
5018: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
5019: // Works like PtInRect but excludes all edges from comparision
5020: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
5021: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5022: function IsFourDigitYear: Boolean;
5023: { moved from JvJVCLUtils }
5024: //Open an object with the shell (url or something like that)
5025: function OpenObject(const Value: string): Boolean; overload;
5026: function OpenObject(Value: PChar): Boolean; overload;
5027: {$IFDEF MSWINDOWS}
5028: //Raise the last Exception
5029: procedure RaiseLastWin32; overload;
5030: procedure RaiseLastWin32(const Text: string); overload;
5031: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
5032: // significant 32 bits of a file's binary version number. Typically, this includes the major and minor
5033: // version placed together in one 32-bit Integer. I
5034: function GetFileVersion(const AFileName: string): Cardinal;
5035: {$EXTERNALSYM GetFileVersion}
5036: //Get version of Shell.dll
5037: function GetShellVersion: Cardinal;
5038: {$EXTERNALSYM GetShellVersion}
5039: // CD functions on HW
5040: procedure OpenCdDrive;
5041: procedure CloseCdDrive;
5042: // returns True if Drive is accessible
5043: function DiskInDrive(Drive: Char): Boolean;
5044: {$ENDIF MSWINDOWS}
5045: //Same as linux function ;
5046: procedure PError(const Text: string);
5047: // execute a program without waiting
5048: procedure Exec(const FileName, Parameters, Directory: string);
5049: // execute a program and wait for it to finish
5050: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5051: // returns True if this is the first instance of the program that is running
5052: function FirstInstance(const ATitle: string): Boolean;
5053: // restores a window based on it's classname and Caption. Either can be left empty
5054: // to widen the search
5055: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5056: // manipulate the traybar and start button
5057: procedure HideTraybar;
```

```

5056: procedure ShowTraybar;
5057: procedure ShowStartButton(Visible: Boolean = True);
5058: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5059: procedure MonitorOn;
5060: procedure MonitorOff;
5061: procedure LowPower;
5062: // send a key to the window named AppName
5063: function SendKey(const AppName: string; Key: Char): Boolean;
5064: {$IFDEF MSWINDOWS}
5065: // returns a list of all win currently visible, the Objects property is filled with their window handle
5066: procedure GetVisibleWindows(List: TStrings);
5067: Function GetVisibleWindowsF( List : TStrings):TStrings';
5068: // associates an extension to a specific program
5069: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5070: procedure AddToRecentDocs(const FileName: string);
5071: function GetRecentDocs: TStringList;
5072: {$ENDIF MSWINDOWS}
5073: function CharIsMoney(const Ch: Char): Boolean;
5074: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5075: function IntToExtended(I: Integer): Extended;
5076: { GetChangedText works out the new text given the current cursor pos & the key pressed
5077: It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5078: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5079: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5080: //function StrIsInteger(const S: string): Boolean;
5081: function StrIsFloatMoney(const Ps: string): Boolean;
5082: function StrIsDateTime(const Ps: string): Boolean;
5083: function PreformatDateString(Ps: string): string;
5084: function BooleanToInteger(const B: Boolean): Integer;
5085: function StringToBoolean(const Ps: string): Boolean;
5086: function SafeStrToDate(const Ps: string): TDateTime;
5087: function SafeStrToDate(const Ps: string): TDateTime;
5088: function SafeStrToTime(const Ps: string): TDateTime;
5089: function StrDelete(const psSub, psMain: string): string;
5090: { returns the fractional value of pcValue }
5091: function TimeOnly(pcValue: TDateTime): TTime;
5092: { returns the integral value of pcValue }
5093: function DateOnly(pcValue: TDateTime): TDate;
5094: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5095: const { TDateTime value used to signify Null value}
5096: NullEquivalentDate: TDateTime = 0.0;
5097: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5098: // Replacement for Win32Check to avoid platform specific warnings in D6
5099: function OSCheck(RetVal: Boolean): Boolean;
5100: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5101: Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5102: not be forced to use FileCtrl unnecessarily }
5103: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5104: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5105: { MinimizeString truncates long string, S, and appends...'symbols,if Length of S is more than MaxLen }
5106: function MinimizeString(const S: string; const MaxLen: Integer): string;
5107: procedure RunDl32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=SW_SHOWDEFAULT);
5108: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion function and calls it, returning major and
5109: minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't be
5110: found. }
5109: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5110: {$ENDIF MSWINDOWS}
5111: procedure ResourceNotFound(ResID: PChar);
5112: function EmptyRect: TRect;
5113: function RectWidth(R: TRect): Integer;
5114: function RectHeight(R: TRect): Integer;
5115: function CompareRect(const R1, R2: TRect): Boolean;
5116: procedure RectNormalize(var R: TRect);
5117: function RectIsSquare(const R: TRect): Boolean;
5118: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5119: //IF AMaxSize = -1, then auto calc Square's max size
5120: {$ENDIF MSWINDOWS}
5121: procedure FreeUnusedOle;
5122: function GetWindowsVersion: string;
5123: function LoadDLL(const LibName: string): THandle;
5124: function RegisterServer(const ModuleName: string): Boolean;
5125: function UnregisterServer(const ModuleName: string): Boolean;
5126: {$ENDIF MSWINDOWS}
5127: { String routines }
5128: function GetEnvVar(const VarName: string): string;
5129: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5130: function StringToPChar(var S: string): PChar;
5131: function StrAlloc(const S: string): PChar;
5132: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5133: function DropT(const S: string): string;
5134: { Memory routines }
5135: function AllocMemo(Size: Longint): Pointer;
5136: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5137: procedure FreeMemeo(var fpBlock: Pointer);
5138: function GetMemeoSize(fpBlock: Pointer): Longint;
5139: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5140: { Manipulate huge pointers routines }
5141: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);

```

```

5142: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5143: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5144: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5145: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5146: function WindowClassName(Wnd: THandle): string;
5147: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5148: procedure ActivateWindow(Wnd: THandle);
5149: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5150: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5151: { SetWindowTop put window to top without recreating window }
5152: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5153: procedure CenterWindow(Wnd: THandle);
5154: function MakeVariant(const Values: array of Variant): Variant;
5155: { Convert dialog units to pixels and backwards }
5156: {$IFDEF MSWINDOWS}
5157: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5158: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5159: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5160: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5161: {$ENDIF MSWINDOWS}
5162: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5163: {$IFDEF BCB}
5164: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): THandle;
5165: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
5166: {$ELSE}
5167: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5168: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5169: {$ENDIF BCB}
5170: {$IFDEF MSWINDOWS}
5171: { BrowseForFolderNative displays Browse For Folder dialog }
5172: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5173: {$ENDIF MSWINDOWS}
5174: procedure AntiAlias(Clip: TBitmap);
5175: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin, XFinal, YFinal: Integer);
5176: procedure CopyRectIBits(ACanvas: TCanvas; const DestRect: TRect;
5177: ABitmap: TBitmap; const SourceRect: TRect);
5178: function IsTrueType(const FontName: string): Boolean;
5179: // Removes all non-numeric characters from AValue and returns the resulting string
5180: function TextToValText(const AValue: string): string;
5181: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString ) : boolean
5182: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings )
5183: Function ReplaceRegExpr( const ARegExpr,AInputStr,
      AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5184: Function QuoteRegExprMetaChars( const AStr : RegExprString ) : RegExprString
5185: Function RegExprSubExpressions( const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean ) :
5186:
5187: *****unit uPSI_JvTFUtils;
5188: Function JExtractYear( ADate : TDateTime ) : Word
5189: Function JExtractMonth( ADate : TDateTime ) : Word
5190: Function JExtractDay( ADate : TDateTime ) : Word
5191: Function ExtractHours( ATime : TDateTime ) : Word
5192: Function ExtractMins( ATIME : TDateTime ) : Word
5193: Function ExtractSecs( ATIME : TDateTime ) : Word
5194: Function ExtractMSecs( ATIME : TDateTime ) : Word
5195: Function FirstOfMonth( ADate : TDateTime ) : TDateTime
5196: Function GetDayOfNthDOW( Year, Month, DOW, N : Word ) : Word
5197: Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer ) : Word
5198: Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer )
5199: Procedure IncDOW( var DOW : TTFFDayOfWeek; N : Integer )
5200: Procedure IncDays( var ADate : TDateTime; N : Integer )
5201: Procedure IncWeeks( var ADate : TDateTime; N : Integer )
5202: Procedure IncMonths( var ADate : TDateTime; N : Integer )
5203: Procedure IncYears( var ADate : TDateTime; N : Integer )
5204: Function EndOfMonth( ADate : TDateTime ) : TDateTime
5205: Function IsFirstOfMonth( ADate : TDateTime ) : Boolean
5206: Function IsEndOfMonth( ADate : TDateTime ) : Boolean
5207: Procedure EnsureMonth( Month : Word )
5208: Procedure EnsureDOW( DOW : Word )
5209: Function EqualDates( D1, D2 : TDateTime ) : Boolean
5210: Function Lesser( N1, N2 : Integer ) : Integer
5211: Function Greater( N1, N2 : Integer ) : Integer
5212: Function GetDivLength( TotalLength, DivCount, DivNum : Integer ) : Integer
5213: Function GetDivNum( TotalLength, DivCount, X : Integer ) : Integer
5214: Function GetDivStart( TotalLength, DivCount, DivNum : Integer ) : Integer
5215: Function DOWToBorl( ADOW : TTFFDayOfWeek ) : Integer
5216: Function BorlToDOW( BorlDOW : Integer ) : TTFFDayOfWeek
5217: Function DateToDOW( ADate : TDateTime ) : TTFFDayOfWeek
5218: Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string )
5219: Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string )
5220: Function JRectWidth( ARect : TRect ) : Integer
5221: Function JRectHeight( ARect : TRect ) : Integer
5222: Function JEmptyRect : TRect
5223: Function IsClassByName( Obj : TObject; ClassName : string ) : Boolean
5224:
5225: procedure SIRegister_MSysUtils(CL: TPPSPascalCompiler);
5226: begin
5227: Procedure HideTaskBarButton( hWindow : HWND )

```

```

5228: Function msLoadStr( ID : Integer ) : String
5229: Function msFormat( fmt : String; params : array of const ) : String
5230: Function msFileExists( const FileName : String ) : Boolean
5231: Function msIntToStr( Int : Int64 ) : String
5232: Function msStrPas( const Str : PChar ) : String
5233: Function msRenamefile( const OldName, NewName : String ) : Boolean
5234: Function CutFileName( s : String ) : String
5235: Function GetVersionInfo( var VersionString : String ) : DWORD
5236: Function FormatTime( t : Cardinal ) : String
5237: Function msCreateDir( const Dir : string ) : Boolean
5238: Function SetAutoRun( NeedAutoRun : Boolean; AppName : String ) : Boolean
5239: Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword ) : DWORD
5240: Function msStrLen( Str : PChar ) : Integer
5241: Function msDirectoryExists( const Directory : String ) : Boolean
5242: Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String ) : String
5243: Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte ) : LongBool
5244: Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
5245: Procedure SetEditWndProc( hWnd : HWND; ptr : TObject )
5246: Function GetTextFromFile( Filename : String ) : string
5247: Function IsTopMost( hWnd : HWND ) : Bool // 'LWA_ALPHA', 'LongWord').SetUIInt( $00000002 );
5248: Function msStrToIntDef( const s : String; const i : Integer ) : Integer
5249: Function msStrToInt( s : String ) : Integer
5250: Function GetItemText( hDlg : THandle; ID : DWORD ) : String
5251: end;
5252:
5253: procedure SIRegister_ESBMaths2(CL: TPSPPascalCompiler);
5254: begin
5255:   //TDynFloatArray', 'array of Extended
5256:   TDynLWordArray', 'array of LongWord
5257:   TDynLIntArray', 'array of LongInt
5258:   TDynFloatMatrix', 'array of TDynFloatArray
5259:   TDynLWordMatrix', 'array of TDynLWordArray
5260:   TDynLIntMatrix', 'array of TDynLIntArray
5261:   Function SquareAll( const X : TDynFloatArray ) : TDynFloatArray
5262:   Function InverseAll( const X : TDynFloatArray ) : TDynFloatArray
5263:   Function LnAll( const X : TDynFloatArray ) : TDynFloatArray
5264:   Function Log10All( const X : TDynFloatArray ) : TDynFloatArray
5265:   Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended ) : TDynFloatArray
5266:   Function AddVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5267:   Function SubVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5268:   Function MultVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5269:   Function DotProduct( const X, Y : TDynFloatArray ) : Extended
5270:   Function MNorm( const X : TDynFloatArray ) : Extended
5271:   Function MatrixIsRectangular( const X : TDynFloatMatrix ) : Boolean
5272:   Procedure MatrixDimensions( const X:TDynFloatMatrix; var Rows,Columns:LongWord; var Rectangular:Boolean );
5273:   Function MatrixIsSquare( const X : TDynFloatMatrix ) : Boolean
5274:   Function MatricesSameDimensions( const X, Y : TDynFloatMatrix ) : Boolean
5275:   Function AddMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix
5276:   Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix )
5277:   Function SubtractMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix
5278:   Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix )
5279:   Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended ) : TDynFloatMatrix
5280:   Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended );
5281:   Function MultiplyMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix;
5282:   Function TransposeMatrix( const X : TDynFloatMatrix ) : TDynFloatMatrix;
5283:   Function GrandMean( const X : TDynFloatMatrix; var N : LongWord ) : Extended
5284: end;
5285:
5286: procedure SIRegister_ESBMaths(CL: TPSPPascalCompiler);
5287: begin
5288:   'ESBMinSingle','Single').setExtended( 1.5e-45 );
5289:   'ESBMaxSingle','Single').setExtended( 3.4e+38 );
5290:   'ESBMinDouble','Double').setExtended( 5.0e-324 );
5291:   'ESBMaxDouble','Double').setExtended( 1.7e+308 );
5292:   'ESBMinExtended','Extended').setExtended( 3.6e-4951 );
5293:   'ESBMaxExtended','Extended').setExtended( 1.1e+4932 );
5294:   'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807 );
5295:   'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807 );
5296:   'ESBSqrt2','Extended').setExtended( 1.4142135623730950488 );
5297:   'ESBSqrt3','Extended').setExtended( 1.7320508075688772935 );
5298:   'ESBSqrt5','Extended').setExtended( 2.2360679774997896964 );
5299:   'ESBSqrt10','Extended').setExtended( 3.1622776601683793320 );
5300:   'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729 );
5301:   'ESBCbrt2','Extended').setExtended( 1.2599210498948731648 );
5302:   'ESBCbrt3','Extended').setExtended( 1.4422495703074083823 );
5303:   'ESBCbrt10','Extended').setExtended( 2.1544346900318837219 );
5304:   'ESBCbrt100','Extended').setExtended( 4.6415888336127788924 );
5305:   'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630 );
5306:   'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440 );
5307:   'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451 );
5308:   'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928 );
5309:   'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695 );
5310:   'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147 );
5311:   'ESBe','Extended').setExtended( 2.7182818284590452354 );
5312:   'ESBe2','Extended').setExtended( 7.3890560989306502272 );
5313:   'ESBePi','Extended').setExtended( 23.140692632779269006 );
5314:   'ESBePiOn2','Extended').setExtended( 4.8104773809653516555 );
5315:   'ESBePiOn4','Extended').setExtended( 2.1932800507380154566 );
5316:   'ESBLn2','Extended').setExtended( 0.69314718055994530942 );

```

```

5317: 'ESBLn10','Extended').setExtended( 2.30258509299404568402);
5318: 'ESBlnPi','Extended').setExtended( 1.14472988584940017414);
5319: 'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5320: 'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5321: 'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5322: 'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5323: 'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5324: 'ESBPi','Extended').setExtended( 3.1415926535897932385);
5325: 'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5326: 'ESBTwopi','Extended').setExtended( 6.2831853071795864769);
5327: 'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5328: 'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5329: 'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5330: 'ESBPiOn2','Extended').setExtended( 1.570796326794896192);
5331: 'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5332: 'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5333: 'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5334: 'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5335: 'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5336: 'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5337: 'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5338: 'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5339: 'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5340: 'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5341: 'ESBLnRt2Pi1','Extended').setExtended( 9.189385332046727E-1);
5342: //LongWord', 'Cardinal
5343: TBitList', 'Word
5344: Function UMul( const Num1, Num2 : LongWord ) : LongWord
5345: Function UMulDiv2p32( const Num1, Num2 : LongWord ) : LongWord
5346: Function UMulDiv( const Num1, Num2, Divisor : LongWord ) : LongWord
5347: Function UMulMod( const Num1, Num2, Modulus : LongWord ) : LongWord
5348: Function SameFloat( const X1, X2 : Extended ) : Boolean
5349: Function FloatIsZero( const X : Extended ) : Boolean
5350: Function FloatIsPositive( const X : Extended ) : Boolean
5351: Function FloatIsNegative( const X : Extended ) : Boolean
5352: Procedure IncLim( var B : Byte; const Limit : Byte )
5353: Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt )
5354: Procedure IncLimW( var B : Word; const Limit : Word )
5355: Procedure IncLimI( var B : Integer; const Limit : Integer )
5356: Procedure IncLimL( var B : LongInt; const Limit : LongInt )
5357: Procedure DecLim( var B : Byte; const Limit : Byte )
5358: Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt )
5359: Procedure DecLimW( var B : Word; const Limit : Word )
5360: Procedure DecLimI( var B : Integer; const Limit : Integer )
5361: Procedure DecLimL( var B : LongInt; const Limit : LongInt )
5362: Function MaxB( const B1, B2 : Byte ) : Byte
5363: Function MinB( const B1, B2 : Byte ) : Byte
5364: Function MaxSI( const B1, B2 : ShortInt ) : ShortInt
5365: Function MinSI( const B1, B2 : ShortInt ) : ShortInt
5366: Function MaxW( const B1, B2 : Word ) : Word
5367: Function MinW( const B1, B2 : Word ) : Word
5368: Function esbMaxI( const B1, B2 : Integer ) : Integer
5369: Function esbMinI( const B1, B2 : Integer ) : Integer
5370: Function MaxL( const B1, B2 : LongInt ) : LongInt
5371: Function MinL( const B1, B2 : LongInt ) : LongInt
5372: Procedure SwapB( var B1, B2 : Byte )
5373: Procedure SwapSI( var B1, B2 : ShortInt )
5374: Procedure SwapW( var B1, B2 : Word )
5375: Procedure SwapI( var B1, B2 : SmallInt )
5376: Procedure SwapL( var B1, B2 : LongInt )
5377: Procedure SwapI32( var B1, B2 : Integer )
5378: Procedure SwapC( var B1, B2 : LongWord )
5379: Procedure SwapInt64( var X, Y : Int64 )
5380: Function esbSign( const B : LongInt ) : ShortInt
5381: Function Max4Word( const X1, X2, X3, X4 : Word ) : Word
5382: Function Min4Word( const X1, X2, X3, X4 : Word ) : Word
5383: Function Max3Word( const X1, X2, X3 : Word ) : Word
5384: Function Min3Word( const X1, X2, X3 : Word ) : Word
5385: Function MaxBArray( const B : array of Byte ) : Byte
5386: Function MaxWArray( const B : array of Word ) : Word
5387: Function MaxSIArray( const B : array of Shortint ) : ShortInt
5388: Function MaxIArray( const B : array of Integer ) : Integer
5389: Function MaxLArray( const B : array of LongInt ) : LongInt
5390: Function MinBArray( const B : array of Byte ) : Byte
5391: Function MinWArray( const B : array of Word ) : Word
5392: Function MinSIArray( const B : array of Shortint ) : ShortInt
5393: Function MinIArray( const B : array of Integer ) : Integer
5394: Function MinLArray( const B : array of Longint ) : Longint
5395: Function SumBArray( const B : array of Byte ) : Byte
5396: Function SumBArray2( const B : array of Byte ) : Word
5397: Function SumSIArray( const B : array of ShortInt ) : ShortInt
5398: Function SumSIArray2( const B : array of ShortInt ) : Integer
5399: Function SumWArray( const B : array of Word ) : Word
5400: Function SumWArray2( const B : array of Word ) : LongInt
5401: Function SumIArray( const B : array of Integer ) : Integer
5402: Function SumLArray( const B : array of LongInt ) : LongInt
5403: Function SumLWArray( const B : array of LongWord ) : LongWord
5404: Function ESBDigits( const X : LongWord ) : Byte
5405: Function BitsHighest( const X : LongWord ) : Integer

```

```

5406: Function ESBitsNeeded( const X : LongWord) : Integer
5407: Function esbGCD( const X, Y : LongWord) : LongWord
5408: Function esbLCM( const X, Y : LongInt) : Int64
5409: //Function esbLCM( const X, Y : LongInt) : LongInt
5410: Function RelativePrime( const X, Y : LongWord) : Boolean
5411: Function Get87ControlWord : TBitList
5412: Procedure Set87ControlWord( const CWord : TBitList)
5413: Procedure SwapExt( var X, Y : Extended)
5414: Procedure SwapDbl( var X, Y : Double)
5415: Procedure SwapSing( var X, Y : Single)
5416: Function esbSgn( const X : Extended) : ShortInt
5417: Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5418: Function ExtMod( const X, Y : Extended) : Extended
5419: Function ExtRem( const X, Y : Extended) : Extended
5420: Function CompMOD( const X, Y : Comp) : Comp
5421: Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5422: Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5423: Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5424: Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5425: Function MaxExt( const X, Y : Extended) : Extended
5426: Function MinExt( const X, Y : Extended) : Extended
5427: Function MaxEArray( const B : array of Extended) : Extended
5428: Function MinEArray( const B : array of Extended) : Extended
5429: Function MaxSArray( const B : array of Single) : Single
5430: Function MinSArray( const B : array of Single) : Single
5431: Function MaxCArray( const B : array of Comp) : Comp
5432: Function MinCArray( const B : array of Comp) : Comp
5433: Function SumSArray( const B : array of Single) : Single
5434: Function SumEArray( const B : array of Extended) : Extended
5435: Function SumSqEArray( const B : array of Extended) : Extended
5436: Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5437: Function SumXYEArray( const X, Y : array of Extended) : Extended
5438: Function SumCArray( const B : array of Comp) : Comp
5439: Function FactorialX( A : LongWord) : Extended
5440: Function PermutationX( N, R : LongWord) : Extended
5441: Function esbBinomialCoeff( N, R : LongWord) : Extended
5442: Function IsPositiveEArray( const X : array of Extended) : Boolean
5443: Function esbGeometricMean( const X : array of Extended) : Extended
5444: Function esbHarmonicMean( const X : array of Extended) : Extended
5445: Function ESBMean( const X : array of Extended) : Extended
5446: Function esbSampleVariance( const X : array of Extended) : Extended
5447: Function esbPopulationVariance( const X : array of Extended) : Extended
5448: Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5449: Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5450: Function GetMedian( const SortedX : array of Extended) : Extended
5451: Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5452: Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5453: Function ESBMagnitude( const X : Extended) : Integer
5454: Function ESBTan( Angle : Extended) : Extended
5455: Function ESB Cot( Angle : Extended) : Extended
5456: Function ESB Cosec( const Angle : Extended) : Extended
5457: Function ESB Sec( const Angle : Extended) : Extended
5458: Function ESB ArcTan( X, Y : Extended) : Extended
5459: Procedure ESB SinCos( Angle : Extended; var SinX, CosX : Extended)
5460: Function ESB ArcCos( const X : Extended) : Extended
5461: Function ESB ArcSin( const X : Extended) : Extended
5462: Function ESB ArcSec( const X : Extended) : Extended
5463: Function ESB ArcCosec( const X : Extended) : Extended
5464: Function ESB Log10( const X : Extended) : Extended
5465: Function ESB Log2( const X : Extended) : Extended
5466: Function ESB LogBase( const X, Base : Extended) : Extended
5467: Function Pow2( const X : Extended) : Extended
5468: Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5469: Function ESB IntPower( const X : Extended; const N : LongInt) : Extended
5470: Function XtoY( const X, Y : Extended) : Extended
5471: Function esbTentToY( const Y : Extended) : Extended
5472: Function esbTwoToY( const Y : Extended) : Extended
5473: Function LogXtoBaseY( const X, Y : Extended) : Extended
5474: Function esbISqr( const I : LongWord) : Longword
5475: Function ILLog2( const I : LongWord) : Longword
5476: Function IGreatestPowerOf2( const N : LongWord) : LongWord
5477: Function ESB ArCosh( X : Extended) : Extended
5478: Function ESB ArSinh( X : Extended) : Extended
5479: Function ESB ArTanh( X : Extended) : Extended
5480: Function ESB Cosh( X : Extended) : Extended
5481: Function ESB Sinh( X : Extended) : Extended
5482: Function ESB Tanh( X : Extended) : Extended
5483: Function InverseGamma( const X : Extended) : Extended
5484: Function esbGamma( const X : Extended) : Extended
5485: Function esbLnGamma( const X : Extended) : Extended
5486: Function esbBeta( const X, Y : Extended) : Extended
5487: Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5488: end;
5489:
5490: *****Integer Huge Cardinal Utils*****
5491: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5492: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5493: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5494: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32

```

```

5495: Function BitCount_8( Value : byte ) : integer
5496: Function BitCount_16( Value : uint16 ) : integer
5497: Function BitCount_32( Value : uint32 ) : integer
5498: Function BitCount_64( Value : uint64 ) : integer
5499: Function CountSetBits_64( Value : uint64 ) : integer TPrimalityTestNoticeProc',
5500: Procedure ( CountPrimalityTests : integer )
5501: Function gcd( a, b : THugeCardinal ) : THugeCardinal
5502: Function lcm( a, b : THugeCardinal ) : THugeCardinal
5503: Function isCoPrime( a, b : THugeCardinal ) : boolean
5504: Function isProbablyPrime(p: THugeCardinal; OnProgress: TProgress; var wasAborted: boolean): boolean
5505: Function hasSmallFactor( p : THugeCardinal ) : boolean
5506: //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
NumbersTested: integer ) : boolean
5507: Function Inverse( Prime, Modulus : THugeCardinal ) : THugeCardinal
5508: Const ('StandardExponent','LongInt'( 65537 );
5509: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
Numbers
5510: Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal ) : boolean')
5511:
5512: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5513: begin
5514: AddTypeS('TXRTLInteger', 'array of Integer
5515: AddClassN(FindClass('TOBJECT'), 'EXRTLMathException
5516: (FindClass('TOBJECT'), 'EXRTLExtendInvalidArgument
5517: AddClassN(FindClass('TOBJECT'), 'EXRTLDivisionByZero
5518: AddClassN(FindClass('TOBJECT'), 'EXRTLExpInvalidArgument
5519: AddClassN(FindClass('TOBJECT'), 'EXRTLInvalidRadix
5520: AddClassN(FindClass('TOBJECT'), 'EXRTLInvalidRadixDigit
5521: AddClassN(FindClass('TOBJECT'), 'EXRTLRootInvalidArgument
5522: 'BitsPerByte','LongInt'( 8 );
5523: BitsPerDigit','LongInt'( 32 );
5524: SignBitMask','LongWord( $80000000 );
5525: Function XRTLAdjustBits( const ABits : Integer ) : Integer
5526: Function XRTLLength( const AInteger : TXRTLInteger ) : Integer
5527: Function XRTLDataBits( const AInteger : TXRTLInteger ) : Integer
5528: Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer )
5529: Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer )
5530: Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer )
5531: Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer ) : Integer
5532: Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer ) : Boolean
5533: Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int16;var AResult:TXRTLInteger):Int
5534: Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5535: Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5536: Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5537: Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger )
5538: Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5539: Procedure XRTLand( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5540: Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5541: Function XRTLSign( const AInteger : TXRTLInteger ) : Integer
5542: Procedure XRTLZero( var AInteger : TXRTLInteger )
5543: Procedure XRTLOne( var AInteger : TXRTLInteger )
5544: Procedure XRTLMOne( var AInteger : TXRTLInteger )
5545: Procedure XRTLTwo( var AInteger : TXRTLInteger )
5546: Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger ) : Integer
5547: Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger ) : Integer
5548: Procedure XRTLFULLsum( const A, B, C : Integer; var Sum, Carry : Integer )
5549: Function XRTLAAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger ) : Integer;
5550: Function XRTLAAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5551: Function XRTLSUB( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger ) : Integer;
5552: Function XRTLSUBL( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5553: Function XRTLCOMPARE( const AInteger1, AInteger2 : TXRTLInteger ) : Integer;
5554: Function XRTLCOMPARE1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64 ) : Integer;
5555: Function XRTLUMUL( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger ) : Integer
5556: Function XRTLMULADD(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5557: Function XRTLMUL( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger ) : Integer
5558: Procedure XRTLDIVMOD( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger )
5559: Procedure XRTLSQR( const AInteger : TXRTLInteger; var AResult : TXRTLInteger )
5560: Procedure XRTLSQRT( const AInteger : TXRTLInteger; var AResult : TXRTLInteger )
5561: Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5562: Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHHighApproxResult:TXRTLInteger)
5563: Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHHighApproxResult:TXRTLInteger);
5564: Procedure XRTLEXP( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5565: Procedure XRTLEXPMOD( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger )
5566: Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5567: Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5568: Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5569: Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5570: Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5571: Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5572: Procedure XRTLSLBLR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5573: Procedure XRTLSABRL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5574: Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5575: Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5576: Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5577: Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)

```

```

5578: Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer ) : string
5579: Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer ) : string
5580: Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer ) : string
5581: Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger )
5582: Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger )
5583: Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer )
5584: Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger );
5585: Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger );
5586: Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger );
5587: Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger )
5588: Procedure XRTLSplit( const AInteger : TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer )
5589: Function XRTLGetMSBIndex( const AInteger : TXRTLInteger ) : Integer
5590: Procedure XRTLMINMax( const AInteger1, AInteger2 : TXRTLInteger; var AMinResult,AMaxResult : TXRTLInteger )
5591: Procedure XRTLMIN( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger );
5592: Procedure XRTLMINl( const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger );
5593: Procedure XRTLMAX( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger );
5594: Procedure XRTLMAXl( const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger );
5595: Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5596: Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger )
5597: Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger )
5598: Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5599: end;
5600:
5601: procedure SIRegister_JvXPCoreUtils(CL: TPSCompiler);
5602: begin
5603:   Function JvXPMethodsEqual( const Method1, Method2 : TMethod ) : Boolean
5604:   Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer )
5605:   Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap );
5606:   Procedure JvXPADJustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect )
5607:   Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLns;const
      AColor:TColor;Rect:TRect;
5608:   Procedure JvXPConvertToGray2( Bitmap : TBitmap )
5609:   Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer )
5610:   Procedure JvXPFRender3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool ;
5611:   Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor )
5612:   Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer )
5613:   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 )
5614: end;
5615:
5616:
5617: procedure SIRegister_uwinstr(CL: TPSCompiler);
5618: begin
5619:   Function StrDec( S : String ) : String
5620:   Function uIsNumeric( var S : String; var X : Float ) : Boolean
5621:   Function ReadNumFromEdit( Edit : TEdit ) : Float
5622:   Procedure WriteNumToFile( var F : Text; X : Float )
5623: end;
5624:
5625: procedure SIRegister_utexplot(CL: TPSCompiler);
5626: begin
5627:   Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean ) : Boolean
5628:   Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean )
5629:   Procedure TeX_LeaveGraphics( Footer : Boolean )
5630:   Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float )
5631:   Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float )
5632:   Procedure TeX_SetGraphTitle( Title : String )
5633:   Procedure TeX_SetOxTitle( Title : String )
5634:   Procedure TeX_SetOyTitle( Title : String )
5635:   Procedure TeX_PlotOxAxis
5636:   Procedure TeX_PlotOyAxis
5637:   Procedure TeX_PlotGrid( Grid : TGrid )
5638:   Procedure TeX_WriteGraphTitle
5639:   Function TeX_SetMaxCurv( NCurv : Byte ) : Boolean
5640:   Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer )
5641:   Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean )
5642:   Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String )
5643:   Procedure TeX_SetCurvStep( CurvIndex, Step : Integer )
5644:   Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer )
5645:   Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer )
5646:   Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer )
5647:   Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean )
5648:   Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix )
5649:   Function Xcm( X : Float ) : Float
5650:   Function Ycm( Y : Float ) : Float
5651: end;
5652:
5653: *-----*)
5654: procedure SIRegister_VarRecUtils(CL: TPSCompiler);
5655: begin
5656:   TConstArray', 'array of TVarRec
5657:   Function CopyVarRec( const Item : TVarRec ) : TVarRec
5658:   Function CreateConstArray( const Elements : array of const ) : TConstArray
5659:   Procedure FinalizeVarRec( var Item : TVarRec )

```

```

5660: Procedure FinalizeConstArray( var Arr : TConstArray)
5661: end;
5662:
5663: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5664: begin
5665:   Function HexBS( B : Byte) : ShortString
5666:   Function HexWS( W : Word) : ShortString
5667:   Function HexLS( L : LongInt) : ShortString
5668:   Function HexPtrS( P : Pointer) : ShortString
5669:   Function BinaryBS( B : Byte) : ShortString
5670:   Function BinaryWS( W : Word) : ShortString
5671:   Function BinaryLS( L : LongInt) : ShortString
5672:   Function OctalBS( B : Byte) : ShortString
5673:   Function OctalWS( W : Word) : ShortString
5674:   Function OctalLS( L : LongInt) : ShortString
5675:   Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5676:   Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5677:   Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5678:   Function Str2Reals( const S : ShortString; var R : Double) : Boolean
5679:   Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5680:   Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5681:   Function Long2StrS( L : LongInt) : ShortString
5682:   Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5683:   Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5684:   Function ValPrepS( const S : ShortString) : ShortString
5685:   Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5686:   Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5687:   Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5688:   Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5689:   Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5690:   Function TrimLeads( const S : ShortString) : ShortString
5691:   Function TrimTrails( const S : ShortString) : ShortString
5692:   Function Trims( const S : ShortString) : ShortString
5693:   Function TrimSpacesS( const S : ShortString) : ShortString
5694:   Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : shortString
5695:   Function Centers( const S : ShortString; Len : Cardinal) : ShortString
5696:   Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5697:   Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5698:   Function ScrambleS( const S, Key : ShortString) : ShortString
5699:   Function Substitutes( const S, FromStr,ToStr : ShortString) : ShortString
5700:   Function Filters( const S, Filters : ShortString) : ShortString
5701:   Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5702:   Function CharCounts( const S : ShortString; C : AnsiChar) : Byte
5703:   Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5704:   Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5705:   Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5706:   Function AsciiCountsS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5707:   Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5708:   Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5709:   Procedure WordWraps(const InSt: ShortString; var OutSt,Overlap: ShortString;
Margin:Cardinal;PadToMargin:Boolean)
5710:   Function CompStringS( const S1, S2 : ShortString) : Integer
5711:   Function CompUCStringS( const S1, S2 : ShortString) : Integer
5712:   Function SoundexS( const S : ShortString) : ShortString
5713:   Function MakeLetterSetS( const S : ShortString) : Longint
5714:   Procedure BMMakeTableS( const MatchString : shortString; var BT : BTable)
5715:   Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;
5716:   Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
Pos:Cardinal):Bool;
5717:   Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5718:   Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5719:   Function JustFilenameS( const PathName : ShortString) : ShortString
5720:   Function JustNameS( const PathName : ShortString) : ShortString
5721:   Function JustExtensionS( const Name : ShortString) : ShortString
5722:   Function JustPathnameS( const PathName : ShortString) : ShortString
5723:   Function AddBackSlashS( const DirName : ShortString) : ShortString
5724:   Function CleanPathNameS( const PathName : ShortString) : ShortString
5725:   Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5726:   Function CommaizeS( L : LongInt) : ShortString
5727:   Function CommaizeChS( L : LongInt; Ch : AnsiChar) : ShortString
5728:   Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
DecPt:Char):ShortString;
5729:   Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
RtCurr:ShortString;Sep:AnsiChar):ShortString;
5730:   Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5731:   Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5732:   Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5733:   Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5734:   Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5735:   Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5736:   Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5737:   Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5738:   Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5739:   Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5740:   Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5741:   Function CopyRights( const S : ShortString; First : Cardinal) : shortString
5742:   Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5743:   Function CopyFromNthWordS(const S,WordDelims:string;const Aword:String;N:Card;var
SubString:ShortString):Bool;

```

```

5744: Function DeleteFromNthWordS( const S, WordDelims: String; AWord: ShortString; N: Card; var
5745: SubStr: ShortString ): Bool;
5746: Function CopyFromToWordS( const S, WordDelims, Word1, Word2: ShortString; N1, N2: Card; var
5747: SubString: ShortString ): Bool;
5748: Function DeleteFromToWords( const S, WordDelims, Wrld1, Wrld2: ShortString; N1, N2: Card; var
5749: SubString: ShortString ): Bool;
5747: Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean ) : ShortString
5748: Function DeleteWithinS( const S, Delimiter : ShortString ) : ShortString
5749: Function ExtractTokensS( const S,
      Delims: ShortString; QuoteChar: AnsiChar; AllowNulls: Boolean; Tokens: TStrings ) : Cardinal
5750: Function IsChAlphaS( C : Char ) : Boolean
5751: Function IsChNumericS( C : Char; const Numbers : ShortString ) : Boolean
5752: Function IsChAlphaNumerics( C : Char; const Numbers : ShortString ) : Boolean
5753: Function IsStrAlphaS( const S : Shortstring ) : Boolean
5754: Function IsStrNumericS( const S, Numbers : ShortString ) : Boolean
5755: Function IsStrAlphaNumericS( const S, Numbers : ShortString ) : Boolean
5756: Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal ) : Boolean
5757: Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal ) : Boolean
5758: Function LastStringS( const S, AString : ShortString; var Position : Cardinal ) : Boolean
5759: Function LeftTrimCharsS( const S, Chars : ShortString ) : ShortString
5760: Function KeepCharsS( const S, Chars : ShortString ) : ShortString
5761: Function RepeatStringS( const RepeatString: ShortString; var Repetitions: Cardinal; MaxLen : Cardinal ) : ShortString;
5762: Function ReplaceStringsS( const S, OldStr, NewStr: ShortString; N: Cardinal; var
      Replacements: Cardinal ) : ShortString;
5763: Function ReplaceStringAllS( const S, OldString, NewString: ShortString; var Replacements: Cardinal ) : ShortString;
5764: Function ReplaceWordS( const S, WordDelims, OldWord, NewW: SString; N: Cardinal; var
      Replacements: Cardinal ) : ShortString
5765: Function ReplaceWordAllS( const S, WordDelims, OldWord, NewWord: ShortString; var
      Replacements: Cardinal ) : ShortString
5766: Function RightTrimCharsS( const S, Chars : ShortString ) : ShortString
5767: Function StrWithinS( const S, SearchStr: ShortString; Start: Cardinal; var Position: Cardinal ) : boolean
5768: Function TrimCharsS( const S, Chars : ShortString ) : ShortString
5769: Function WordPosS( const S, WordDelims, AWord: ShortString; N: Cardinal; var Position: Cardinal ) : Boolean
5770: end;
5771:
5772:
5773: *****unit uPSI_StUtils; from Systools4*****
5774: Function SignL( L : LongInt ) : Integer
5775: Function SignF( F : Extended ) : Integer
5776: Function MinWord( A, B : Word ) : Word
5777: Function MidWord( W1, W2, W3 : Word ) : Word
5778: Function MaxWord( A, B : Word ) : Word
5779: Function MinLong( A, B : LongInt ) : LongInt
5780: Function MidLong( L1, L2, L3 : LongInt ) : LongInt
5781: Function MaxLong( A, B : LongInt ) : LongInt
5782: Function MinFloat( F1, F2 : Extended ) : Extended
5783: Function MidFloat( F1, F2, F3 : Extended ) : Extended
5784: Function MaxFloat( F1, F2 : Extended ) : Extended
5785: Function MakeInteger16( H, L : Byte ) : SmallInt
5786: Function MakeWordS( H, L : Byte ) : Word
5787: Function SwapNibble( B : Byte ) : Byte
5788: Function SwapWord( L : LongInt ) : LongInt
5789: Procedure SetFlag( var Flags : Word; FlagMask : Word )
5790: Procedure ClearFlag( var Flags : Word; FlagMask : Word )
5791: Function FlagIsSet( Flags, FlagMask : Word ) : Boolean
5792: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte )
5793: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte )
5794: Function ByteFlagIsSet( Flags, FlagMask : Byte ) : Boolean
5795: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt )
5796: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt )
5797: Function LongFlagIsSet( Flags, FlagMask : LongInt ) : Boolean
5798: Procedure ExchangeBytes( var I, J : Byte )
5799: Procedure ExchangeWords( var I, J : Word )
5800: Procedure ExchangeLongInts( var I, J : LongInt )
5801: Procedure ExchangeStructs( var I, J, Size : Cardinal )
5802: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word )
5803: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal )
5804: Function AddWordToPtr( P : __Pointer; W : Word ) : __Pointer
5805: //*****uPSI_StFIN;*****
5806: Function AccruedInterestMaturity( Issue, Maturity: TStDate; Rate, Par: Extended; Basis: TStBasis ) : Extended
5807: Function AccruedInterestPeriodic( Issue, Settlement, Maturity: TStDate; Rate,
      Par: Extended; Frequency: TStFrequency; Basis : TStBasis ) : Extended
5808: Function BondDuration( Settlement, Maturity: TStDate; Rate,
      Yield: Ext; Frequency: TStFrequency; Basis: TStBasis ) : Extended;
5809: Function BondPrice( Settlement, Maturity: TStDate; Rate, Yield, Redempt: Ext; Freq: TStFrequency; Basis: TStBasis ) :
      Extended
5810: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5811: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5812: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis ) : LongInt
5813: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer ) : Extended
5814: Function DiscountRate( Settlement, Maturity: TStDate; Price, Redemption: Extended; Basis: TStBasis ) : Extended;
5815: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer ) : Extended
5816: Function DollarToDecimalText( DecDollar : Extended; Fraction : Integer ) : string
5817: Function DollarToFraction( DecDollar : Extended; Fraction : Integer ) : Extended
5818: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer ) : string
5819: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency ) : Extended

```

```

5820: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
5821: PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5822: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double ) : Extended
5823: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer ) : Extended
5824: Function InterestRateS(NPeriods:Int;Pmt,PV,
5825: FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extended;
5826: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended ) : Extended
5827: Function IsCardValid( const S : string ) : Boolean
5828: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
5829: Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5830: Function ModifiedIRR(const Values : array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5831: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5832: Function NetPresentValue( Rate : Extended; const Values : array of Double ) : Extended
5833: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency ) : Extended
5834: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5835: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
5836: : TStPaymentTime): Extended
5837: Function PresentValueS( Rate : Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
5838: Timing: TStPaymentTime): Extended
5839: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5840: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean ) : Extended
5841: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5842: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5843: Function TBillyard( Settlement, Maturity : TStDate; Price : Extended ) : Extended
5844: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
5845: Factor : Extended; NoSwitch : boolean ) : Extended
5846: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5847: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
5848: Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5849: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray )
5850: //*****unit unit uPSI_StStat; Statistic Package of SysTools*****
5851: Function AveDev( const Data : array of Double ) : Double
5852: Function AveDev16( const Data, NData : Integer ) : Double
5853: Function Confidence( Alpha, StandardDev : Double; Size : LongInt ) : Double
5854: Function Correlation( const Data1, Data2 : array of Double ) : Double
5855: Function Correlation16( const Data1, Data2, NData : Integer ) : Double
5856: Function Covariance( const Data1, Data2 : array of Double ) : Double
5857: Function Covariance16( const Data1, Data2, NData : Integer ) : Double
5858: Function DevSq( const Data : array of Double ) : Double
5859: Function DevSq16( const Data, NData : Integer ) : Double
5860: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5861: //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5862: Function GeometricMeanS( const Data : array of Double ) : Double
5863: Function GeometricMean16( const Data, NData : Integer ) : Double
5864: Function HarmonicMeanS( const Data : array of Double ) : Double
5865: Function HarmonicMean16( const Data, NData : Integer ) : Double
5866: Function Largest( const Data : array of Double; K : Integer ) : Double
5867: Function Largest16( const Data, NData : Integer; K : Integer ) : Double
5868: Function MedianS( const Data : array of Double ) : Double
5869: Function Median16( const Data, NData : Integer ) : Double
5870: Function Mode( const Data : array of Double ) : Double
5871: Function Mode16( const Data, NData : Integer ) : Double
5872: Function Percentile( const Data : array of Double; K : Double ) : Double
5873: Function Percentile16( const Data, NData : Integer; K : Double ) : Double
5874: Function PercentRank( const Data : array of Double; X : Double ) : Double
5875: Function PercentRank16( const Data, NData : Integer; X : Double ) : Double
5876: Function Permutations( Number, NumberChosen : Integer ) : Extended
5877: Function Combinations( Number, NumberChosen : Integer ) : Extended
5878: Function Factorials( N : Integer ) : Extended
5879: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean ) : Integer
5880: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean ) : Integer
5881: Function Smallest( const Data : array of Double; K : Integer ) : Double
5882: Function Smallest16( const Data, NData : Integer; K : Integer ) : Double
5883: Function TrimMean( const Data : array of Double; Percent : Double ) : Double
5884: Function TrimMean16( const Data, NData : Integer; Percent : Double ) : Double
5885: AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5886: +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5887: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
5888: LF:TStLinEst;ErrorStats:Bool;
5889: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
5890: LF:TStLinEst;ErrorStats:Bool;
5891: Function Forecast(X:Double;const KnownY:array of Double;const KnownX: array of Double) : Double
5892: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5893: Function Intercept( const KnownY : array of Double; const KnownX : array of Double ) : Double
5894: Function RSquared( const KnownY : array of Double; const KnownX : array of Double ) : Double
5895: Function Slope( const KnownY : array of Double; const KnownX : array of Double ) : Double
5896: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double ) : Double
5897: Function BetaDist( Probability, Alpha, Beta, A, B : Single ) : Single
5898: Function BinomDist( Numbers, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean ) : Single
5899: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single ) : Integer
5900: Function ChiDist( X : Single; DegreesFreedom : Integer ) : Single

```

```

5900: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5901: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5902: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5903: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5904: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5905: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5906: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5907: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5908: Function NormSDist( Z : Single) : Single
5909: Function NormSInv( Probability : Single) : Single
5910: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5911: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5912: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5913: Function Erfc( X : Single) : Single
5914: Function GammaLn( X : Single) : Single
5915: Function LargestSort( const Data : array of Double; K : Integer) : Double
5916: Function SmallestSort( const Data : array of double; K : Integer) : Double
5917:
5918: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5919: Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5920: Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5921: Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5922: Function DefaultMergeName( MergeNum : Integer) : string
5923: Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5924:
5925: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5926: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5927: Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5928: Function SunPos( UT : TStDateTimeRec) : TStPosRec
5929: Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5930: Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5931: Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5932: Function LunarPhase( UT : TStDateTimeRec) : Double
5933: Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5934: Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5935: Function FirstQuarter( D : TStDate) : TStLunarRecord
5936: Function FullMoon( D : TStDate) : TStLunarRecord
5937: Function LastQuarter( D : TStDate) : TStLunarRecord
5938: Function NewMoon( D : TStDate) : TStLunarRecord
5939: Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5940: Function NextFullMoon( D : TStDate) : TStDateTimeRec
5941: Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5942: Function NextNewMoon( D : TStDate) : TStDateTimeRec
5943: Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5944: Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5945: Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5946: Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5947: Function SiderealTime( UT : TStDateTimeRec) : Double
5948: Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5949: Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5950: Function SEaster( Y, Epoch : Integer) : TStDate
5951: Function DateTimeToAJD( D : TDateTime) : Double
5952: Function HoursMin( RA : Double) : ShortString
5953: Function DegsMin( DC : Double) : ShortString
5954: Function AJDToDate( D : Double) : TDateTime
5955:
5956: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5957: Function CurrentDate : TStDate
5958: Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5959: Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5960: Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5961: Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5962: Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5963: Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5964: Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5965: Function WeekOffYear( Julian : TStDate) : Byte
5966: Function AstJulianDate( Julian : TStDate) : Double
5967: Function AstJulianDateToStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5968: Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5969: Function StDayOfWeek( Julian : TStDate) : TStDayType
5970: Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5971: Function StIsLeapYear( Year : Integer) : Boolean
5972: Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5973: Function ResolveEpoch( Year, Epoch : Integer) : Integer
5974: Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5975: Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5976: Function HMSToStTime( Hours, Minutes, Seconds : Byte) : TStTime
5977: Function CurrentTime : TStTime
5978: Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5979: Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5980: Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5981: Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5982: Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5983: Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt)
5984: Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5985: Function DateTimeToStDate( DT : TDateTime) : TStDate
5986: Function DateTimeToStTime( DT : TDateTime) : TStTime
5987: Function StDateToDate( D : TStDate) : TDateTime
5988: Function StTimeToDate( T : TStTime) : TDateTime

```

```

5989: Function Convert2ByteDate( TwoByteDate : Word ) : TStDate
5990: Function Convert4ByteDate( FourByteDate : TStDate ) : Word
5991:
5992: Procedure SIRegister_StDateSt(CL: TPPascalCompiler);
5993: Function DateStringHMSToAstJD( const Picture, DS : string; H, M, S, Epoch : integer ) : Double
5994: Function MonthToString( const Month : Integer ) : string
5995: Function DateStringToStDate( const Picture, S : string; Epoch : Integer ) : TStDate
5996: Function DateStringToDMY( const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5997: Function StDateToString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5998: Function DayOfWeekToString( const WeekDay : TStDayType) : string
5999: Function DMYToDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string;
6000: Function CurrentDateString( const Picture : string; Pack : Boolean) : string
6001: Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
6002: Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
6003: Function TimeStringToStTime( const Picture, S : string) : TStTime
6004: Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
6005: Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
6006: Function DatestringIsBlank( const Picture, S : string ) : Boolean
6007: Function InternationalDate( ForceCentury : Boolean ) : string
6008: Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean ) : string
6009: Function InternationalTime( ShowSeconds : Boolean ) : string
6010: Procedure ResetInternationalInfo
6011:
6012: procedure SIRegister_StBase(CL: TPPascalCompiler);
6013: Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer ) : Boolean
6014: Function AnsiUpperCaseShort32( const S : string ) : string
6015: Function AnsiCompareTextShort32( const S1, S2 : string ) : Integer
6016: Function AnsiCompareStrShort32( const S1, S2 : string ) : Integer
6017: Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer ) : Longint
6018: Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
6019: Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte )
6020: Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal )
6021: Function Upcase( C : AnsiChar ) : AnsiChar
6022: Function LoCase( C : AnsiChar ) : AnsiChar
6023: Function CompareLetterSets( Set1, Set2 : LongInt ) : Cardinal
6024: Function CompStruct( const S1, S2, Size : Cardinal ) : Integer
6025: Function Search( const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
6026: Function StSearch( const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6027: Function SearchUC( const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6028: Function IsOrInheritsFrom( Root, Candidate : TClass ) : boolean
6029: Procedure RaiseContainerError( Code : longint )
6030: Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const )
6031: Function ProductOverflow( A, B : LongInt ) : Boolean
6032: Function StNewStr( S : string ) : PShortString
6033: Procedure StDisposeStr( PS : PShortString )
6034: Procedure VallLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer )
6035: Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer )
6036: Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer )
6037: Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt )
6038: Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt )
6039: Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string )
6040:
6041: procedure SIRegister_usvd(CL: TPPascalCompiler);
6042: begin
6043:   Procedure SV_DecomP( A : TMatrix; Lb, Ubl, Ub2 : Integer; S : TVector; V : TMatrix )
6044:   Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float )
6045:   Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ubl,Ub2:Integer;X:TVector);
6046:   Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ubl, Ub2 : Integer; A : TMatrix )
6047:   Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6048: end;
6049:
6050: //*****unit unit ; StMath Package of SysTools*****
6051: Function IntPowerS( Base : Extended; Exponent : Integer ) : Extended
6052: Function PowerS( Base, Exponent : Extended ) : Extended
6053: Function StInvCos( X : Double ) : Double
6054: Function StInvSin( Y : Double ) : Double
6055: Function StInvTan2( X, Y : Double ) : Double
6056: Function StTan( A : Double ) : Double
6057: Procedure DumpException; //unit StExpEng;
6058: Function HexifyBlock( var Buffer, BufferSize : Integer ) : string
6059:
6060: //*****unit unit ; STCRC Package of SysTools*****
6061: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
6062: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
6063: Function Adler32OfFile( FileName : AnsiString ) : LongInt
6064: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
6065: Function Crc16OfFile( Stream : TStream; CurCrc : Cardinal ) : Cardinal
6066: Function Crc16OfFile( FileName : AnsiString ) : Cardinal
6067: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
6068: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
6069: Function Crc32OfFile( FileName : AnsiString ) : LongInt
6070: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
6071: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
6072: Function InternetSumOfFile( FileName : AnsiString ) : Cardinal
6073: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
6074: Function Kermit16OfFile( Stream : TStream; CurCrc : Cardinal ) : Cardinal
6075: Function Kermit16OfFile( FileName : AnsiString ) : Cardinal
6076:
6077: //*****unit unit ; StBCD Package of SysTools*****

```

```

6078: Function AddBcd( const B1, B2 : Tbcds ) : Tbcds
6079: Function SubBcd( const B1, B2 : Tbcds ) : Tbcds
6080: Function MulBcd( const B1, B2 : Tbcds ) : Tbcds
6081: Function DivBcd( const B1, B2 : Tbcds ) : Tbcds
6082: Function ModBcd( const B1, B2 : Tbcds ) : Tbcds
6083: Function NegBcd( const B : Tbcds ) : Tbcds
6084: Function AbsBcd( const B : Tbcds ) : Tbcds
6085: Function FracBcd( const B : Tbcds ) : Tbcds
6086: Function IntBcd( const B : Tbcds ) : Tbcds
6087: Function RoundDigitsBcd( const B : Tbcds; Digits : Cardinal ) : Tbcds
6088: Function RoundPlacesBcd( const B : Tbcds; Places : Cardinal ) : Tbcds
6089: Function ValBcd( const S : string ) : Tbcds
6090: Function LongBcd( L : LongInt ) : Tbcds
6091: Function ExtBcd( E : Extended ) : Tbcds
6092: Function ExpBcd( const B : Tbcds ) : Tbcds
6093: Function LnBcd( const B : Tbcds ) : Tbcds
6094: Function IntPowBcd( const B : Tbcds; E : LongInt ) : Tbcds
6095: Function PowBcd( const B, E : Tbcds ) : Tbcds
6096: Function SqrtBcd( const B : Tbcds ) : Tbcds
6097: Function CmpBcd( const B1, B2 : Tbcds ) : Integer
6098: Function EqDigitsBcd( const B1, B2 : Tbcds; Digits : Cardinal ) : Boolean
6099: Function EqPlacesBcd( const B1, B2 : Tbcds; Digits : Cardinal ) : Boolean
6100: Function IsIntBcd( const B : Tbcds ) : Boolean
6101: Function TruncBcd( const B : Tbcds ) : LongInt
6102: Function BcdExt( const B : Tbcds ) : Extended
6103: Function RoundBcd( const B : Tbcds ) : LongInt
6104: Function StrBcd( const B : Tbcds; Width, Places : Cardinal ) : string
6105: Function StrExpBcd( const B : Tbcds; Width : Cardinal ) : string
6106: Function FormatBcd( const Format : string; const B : Tbcds ) : string
6107: Function StrGeneralBcd( const B : Tbcds ) : string
6108: Function FloatFormBcd(const Mask:string;B:Tbcds;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6109: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte )
6110:
6111: //*****unit unit ; StTxtData; TStTextDataRecordSet Package of SysTools*****
6112: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings )
6113: Function StFieldTypeToStr( FieldType : TStSchemaFieldType ) : AnsiString
6114: Function StStrToFieldType( const S : AnsiString ) : TStSchemaFieldType
6115: Function StDeEscape( const EscStr : AnsiString ) : Char
6116: Function StDoEscape( Delim : Char ) : AnsiString
6117: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char ) : AnsiString
6118: Function AnsiHashText( const S : string; Size : Integer ) : Integer
6119: Function AnsiHashStr( const S : string; Size : Integer ) : Integer
6120: Function AnsiELFHashText( const S : string; Size : Integer ) : Integer
6121: Function AnsiELFHashStr( const S : string; Size : Integer ) : Integer
6122:
6123: //*****unit unit ; StNetCon Package of SysTools*****
6124: with AddClassN(FindClass('TStComponent'), 'TStNetConnection') do begin
6125:   Constructor Create( AOwner : TComponent )
6126:   Function Connect : DWord
6127:   Function Disconnect : DWord
6128:   RegisterProperty('Password', 'String', iptrw);
6129:   Property('UserName', 'String', iptrw);
6130:   Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6131:   Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6132:   Property('LocalDevice', 'String', iptrw);
6133:   Property('ServerName', 'String', iptrw);
6134:   Property('ShareName', 'String', iptrw);
6135:   Property('OnConnect', 'TNotifyEvent', iptrw);
6136:   Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6137:   Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6138:   Property('OnDisconnect', 'TNotifyEvent', iptrw);
6139:   Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6140:   Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6141: end;
6142: //*****Thread Functions Context of Win API --- more objects in SyncObjs.pas
6143: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6144: Procedure InitializeCriticalSection( var lpCriticalSection : TRTCriticalSection )
6145: Procedure EnterCriticalSection( var lpCriticalSection : TRTCriticalSection )
6146: Procedure LeaveCriticalSection( var lpCriticalSection : TRTCriticalSection )
6147: Function InitializeCriticalSectionAndSpinCount(var
6148:   lpCriticalSection:TRTCriticalSection;dwSpinCount:DWORD):BOOL;
6149: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTCriticalSection;dwSpinCount:DWORD):DWORD;
6150: Procedure TryEnterCriticalSection( var lpCriticalSection : TRTCriticalSection ) : BOOL
6151: Procedure DeleteCriticalSection( var lpCriticalSection : TRTCriticalSection )
6152: Function GetThreadContext( hThread : THandle; var lpContext : TContext ) : BOOL
6153: Function SetThreadContext( hThread : THandle; const lpContext : TContext ) : BOOL
6154: Function SuspendThread( hThread : THandle ) : DWORD
6155: Function ResumeThread( hThread : THandle ) : DWORD
6156: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6157: Procedure GetCurrentThread : THandle
6158: Function ExitThread( dwExitCode : DWORD )
6159: Function TerminateThread( hThread : THandle; dwExitCode : DWORD ) : BOOL
6160: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD ) : BOOL
6161: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD ) : DWORD
6162: Function MakeProcInstance( Proc : FARPROC; Instance : THandle ) : FARPROC
6163: Procedure FreeProcInstance( Proc : FARPROC )
6164: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD )
6165: Function DisableThreadLibraryCalls( hLibModule : HMODULE ) : BOOL

```

```

6166: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6167: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6168: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool);
6169: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6170: Function CreateParallelJob(ASelf: TObject; ATarget: Pointer; AParam: Ptr; ASafeSection: bool; TParallelJob);
6171: Function CreateParallelJob1(ATarget: Pointer; AParam: Pointer; ASafeSection: boolean) : TParallelJob;
6172: Function CurrentParallelJobInfo : TParallelJobInfo;
6173: Function ObtainParallelJobInfo : TParallelJobInfo;
6174: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo );
6175: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD ) : BOOL';
6176: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle ) : BOOL';
6177: Function
  DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
  TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL';
6178: Function SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL';
6179: Function DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
  lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL; dwOptions : DWORD ) : BOOL';
6180: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD ) : BOOL';
6181: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD ) : BOOL';
6182:
6183: *****unit uPSI_JclMime;
6184: Function MimeEncodeString( const S : AnsiString ) : AnsiString;
6185: Function MimeDecodeString( const S : AnsiString ) : AnsiString;
6186: Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream );
6187: Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream );
6188: Function MimeEncodedSize( const I : Cardinal ) : Cardinal;
6189: Function MimeDecodedSize( const I : Cardinal ) : Cardinal;
6190: Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer );
6191: Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6192: Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
  OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal;
6193: Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):Cardinal;
6194:
6195: *****unit uPSI_JclPrint;
6196: Procedure DirectPrint( const Printer, Data : string );
6197: Procedure SetPrinterPixelsPerInch;
6198: Function GetPrinterResolution : TPoint;
6199: Function CharFitsWithinDots( const Text : string; const Dots : Integer ) : Integer;
6200: Procedure PrintMemo( const Memo : TMemo; const Rect : TRect );
6201:
6202:
6203: //*****unit uPSI_ShLwApi;*****
6204: Function StrChr( lpStart : PChar; wMatch : WORD ) : PChar;
6205: Function StrChrI( lpStart : PChar; wMatch : WORD ) : PChar;
6206: Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer ) : Integer;
6207: Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer ) : Integer;
6208: Function StrCSpn( lpStr_ , lpSet : PChar ) : Integer;
6209: Function StrCSpnI( lpStr1, lpSet : PChar ) : Integer;
6210: Function StrDup( lpSrch : PChar ) : PChar;
6211: Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT ) : PChar;
6212: Function StrFormatKBSIZE( qdw : Dword; szBuf : PChar; uiBufSize : UINT ) : PChar;
6213: Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer;
6214: Function StrIsIntLEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer ) : BOOL;
6215: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer ) : PChar;
6216: Function StrPBrk( psz, pszSet : PChar ) : PChar;
6217: Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD ) : PChar;
6218: Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD ) : PChar;
6219: Function StrRStrI( lpSource, lpLast, lpSrch : PChar ) : PChar;
6220: Function StrSpn( psz, pszSet : PChar ) : Integer;
6221: Function StrStr( lpFirst, lpSrch : PChar ) : PChar;
6222: Function StrStrI( lpFirst, lpSrch : PChar ) : PChar;
6223: Function StrToInt( lpSrch : PChar ) : Integer;
6224: Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer ) : BOOL;
6225: Function StrTrim( psz : PChar; pszTrimChars : PChar ) : BOOL;
6226: Function ChrCmpI( w1, w2 : WORD ) : BOOL;
6227: Function ChrCmpIA( w1, w2 : WORD ) : BOOL;
6228: Function ChrCmpIW( w1, w2 : WORD ) : BOOL;
6229: Function StrIntlEqN( s1, s2 : PChar; nChar : Integer ) : BOOL;
6230: Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer ) : BOOL;
6231: Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer ) : PChar;
6232: Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer ) : PChar;
6233: Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer ) : BOOL;
6234: Function IntlStrEqN( s1, s2 : PChar; nChar : Integer ) : BOOL;
6235: SZ_CONTENTTYPE_HTML', 'String 'text/html
6236: SZ_CONTENTTYPE_HTMLW', 'String 'text/html
6237: SZ_CONTENTTYPE_HTML', 'string SZ_CONTENTTYPE_HTML);
6238: SZ_CONTENTTYPE_CDF', 'String 'application/x-cdf
6239: SZ_CONTENTTYPE_CDFW', 'String 'application/x-cdf
6240: SZ_CONTENTTYPE_CDF', 'string SZ_CONTENTTYPE_CDF);
6241: Function PathIsHTMLFile( pszPath : PChar ) : BOOL;
6242: STIF_DEFAULT', 'LongWord( $00000000);
6243: STIF_SUPPORT_HEX', 'LongWord( $00000001);
6244: Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer ) : Integer;
6245: Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer ) : PChar;
6246: Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer ) : PChar;
6247: Function PathAddBackslash( pszPath : PChar ) : PChar;
6248: Function PathAddExtension( pszPath : PChar; pszExt : PChar ) : BOOL;
6249: Function PathAppend( pszPath : PChar; pMore : PChar ) : BOOL;

```

```

6250: Function PathBuildRoot( szRoot : PChar; iDrive : Integer ) : PChar
6251: Function PathCanonicalize( pszBuf : PChar; pszPath : PChar ) : BOOL
6252: Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar ) : PChar
6253: Function PathCompactPath( hdc : HDC; pszPath : PChar; dx : UINT ) : BOOL
6254: Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD ) : BOOL
6255: Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar ) : Integer
6256: Function PathFileExists( pszPath : PChar ) : BOOL
6257: Function PathFindExtension( pszPath : PChar ) : PChar
6258: Function PathFindFileName( pszPath : PChar ) : PChar
6259: Function PathFindNextComponent( pszPath : PChar ) : PChar
6260: Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar ) : BOOL
6261: Function PathGetArgs( pszPath : PChar ) : PChar
6262: Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6263: Function PathIsLFNfileSpec( lpName : PChar ) : BOOL
6264: Function PathGetCharType( ch : Char ) : UINT
6265: GCT_INVALID', LongWord( $0000 );
6266: GCT_LFNCHAR', LongWord( $0001 );
6267: GCT_SHORTCHAR', LongWord( $0002 );
6268: GCT_WILD', LongWord( $0004 );
6269: GCT_SEPARATOR', LongWord( $0008 );
6270: Function PathGetDriveNumber( pszPath : PChar ) : Integer
6271: Function PathIsDirectory( pszPath : PChar ) : BOOL
6272: Function PathIsDirectoryEmpty( pszPath : PChar ) : BOOL
6273: Function PathIsFileSpec( pszPath : PChar ) : BOOL
6274: Function PathIsPrefix( pszPrefix, pszPath : PChar ) : BOOL
6275: Function PathIsRelative( pszPath : PChar ) : BOOL
6276: Function PathIsRoot( pszPath : PChar ) : BOOL
6277: Function PathIsSameRoot( pszPath1, pszPath2 : PChar ) : BOOL
6278: Function PathIsUNC( pszPath : PChar ) : BOOL
6279: Function PathIsNetworkPath( pszPath : PChar ) : BOOL
6280: Function PathIsUNCServer( pszPath : PChar ) : BOOL
6281: Function PathIsUNCServerShare( pszPath : PChar ) : BOOL
6282: Function PathIsContentType( pszPath, pszContentType : PChar ) : BOOL
6283: Function PathIsURL( pszPath : PChar ) : BOOL
6284: Function PathMakePretty( pszPath : PChar ) : BOOL
6285: Function PathMatchSpec( pszFile, pszSpec : PChar ) : BOOL
6286: Function PathParseIconLocation( pszIconFile : PChar ) : Integer
6287: Procedure PathQuoteSpaces( lpsz : PChar )
6288: Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6289: Procedure PathRemoveArgs( pszPath : PChar )
6290: Function PathRemoveBackslash( pszPath : PChar ) : PChar
6291: Procedure PathRemoveBlanks( pszPath : PChar )
6292: Procedure PathRemoveExtension( pszPath : PChar )
6293: Function PathRemoveFileSpec( pszPath : PChar ) : BOOL
6294: Function PathRenameExtension( pszPath : PChar; pszExt : PChar ) : BOOL
6295: Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6296: Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar )
6297: Function PathSkipRoot( pszPath : PChar ) : PChar
6298: Procedure PathStripPath( pszPath : PChar )
6299: Function PathStripToRoot( pszPath : PChar ) : BOOL
6300: Procedure PathUnquoteSpaces( lpsz : PChar )
6301: Function PathMakeSystemFolder( pszPath : PChar ) : BOOL
6302: Function PathUnmakeSystemFolder( pszPath : PChar ) : BOOL
6303: Function PathIsSystemFolder( pszPath : PChar; dwAttrib : DWORD ) : BOOL
6304: Procedure PathUndecorate( pszPath : PChar )
6305: Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6306: URL_SCHEME_INVALID', LongInt'( - 1 );
6307: URL_SCHEME_UNKNOWN', LongInt'( 0 );
6308: URL_SCHEME_FTP', LongInt'( 1 );
6309: URL_SCHEME_HTTP', LongInt'( 2 );
6310: URL_SCHEME_GOPHER', LongInt'( 3 );
6311: URL_SCHEME_MAILTO', LongInt'( 4 );
6312: URL_SCHEME_NEWS', LongInt'( 5 );
6313: URL_SCHEME_NNTP', LongInt'( 6 );
6314: URL_SCHEME_TELNET', LongInt'( 7 );
6315: URL_SCHEME_WAIS', LongInt'( 8 );
6316: URL_SCHEME_FILE', LongInt'( 9 );
6317: URL_SCHEME_MK', LongInt'( 10 );
6318: URL_SCHEME_HTTPS', LongInt'( 11 );
6319: URL_SCHEME_SHELL', LongInt'( 12 );
6320: URL_SCHEME_SNEWS', LongInt'( 13 );
6321: URL_SCHEME_LOCAL', LongInt'( 14 );
6322: URL_SCHEME_JAVASCRIPT', LongInt'( 15 );
6323: URL_SCHEME_VBSCRIPT', LongInt'( 16 );
6324: URL_SCHEME_ABOUT', LongInt'( 17 );
6325: URL_SCHEME_RES', LongInt'( 18 );
6326: URL_SCHEME_MAXVALUE', LongInt'( 19 );
6327: URL_SCHEME', Integer
6328: URL_PART_NONE', LongInt'( 0 );
6329: URL_PART_SCHEME', LongInt'( 1 );
6330: URL_PART_HOSTNAME', LongInt'( 2 );
6331: URL_PART_USERNAME', LongInt'( 3 );
6332: URL_PART_PASSWORD', LongInt'( 4 );
6333: URL_PART_PORT', LongInt'( 5 );
6334: URL_PART_QUERY', LongInt'( 6 );
6335: URL_PART', DWORD
6336: URLIS_URL', LongInt'( 0 );
6337: URLIS_OPAQUE', LongInt'( 1 );
6338: URLIS_NOHISTORY', LongInt'( 2 );

```

```

6339: URLIS_FILEURL', 'LongInt'( 3 );
6340: URLIS_APPLICABLE', 'LongInt'( 4 );
6341: URLIS_DIRECTORY', 'LongInt'( 5 );
6342: URLIS_HASQUERY', 'LongInt'( 6 );
6343: TUrlIs', 'DWORD
6344: URL_UNESCAPE', 'LongWord( $10000000 );
6345: URL_ESCAPE_UNSAFE', 'LongWord( $20000000 );
6346: URL_PLUGGABLE_PROTOCOL', 'LongWord( $40000000 );
6347: URL_WININET_COMPATIBILITY', 'LongWord( DWORD( $80000000 ) );
6348: URL_DONT_ESCAPE_EXTRA_INFO', 'LongWord( $02000000 );
6349: URL_ESCAPE_SPACES_ONLY', 'LongWord( $04000000 );
6350: URL_DONT_SIMPLIFY', 'LongWord( $08000000 );
6351: URL_NO_META', 'longword( URL_DONT_SIMPLIFY );
6352: URL_UNESCAPE_INPLACE', 'LongWord( $00100000 );
6353: URL_CONVERT_IF_DOSPATH', 'LongWord( $00200000 );
6354: URL_UNESCAPE_HIGH_ANSI_ONLY', 'LongWord( $00400000 );
6355: URL_INTERNAL_PATH', 'LongWord( $00800000 );
6356: URL_FILE_USE_PATHURL', 'LongWord( $00010000 );
6357: URL_ESCAPE_PERCENT', 'LongWord( $00001000 );
6358: URL_ESCAPE_SEGMENT_ONLY', 'LongWord( $00002000 );
6359: URL_PARTFLAG_KEEPSCHEME', 'LongWord( $00000001 );
6360: URL_APPLY_DEFAULT', 'LongWord( $00000001 );
6361: URL_APPLY_GUESSSCHEME', 'LongWord( $00000002 );
6362: URL_APPLY_GUESSFILE', 'LongWord( $00000004 );
6363: URL_APPLY_FORCEAPPLY', 'LongWord( $00000008 );
6364: Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL ) : Integer
6365: Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD) : HRESULT;
6366: Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD) : HRESULT;
6367: Function UrlIsOpaque( pszURL : PChar ) : BOOL
6368: Function UrlIsNoHistory( pszURL : PChar ) : BOOL
6369: Function UrlIsFileUrl( pszURL : PChar ) : BOOL
6370: Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs ) : BOOL
6371: Function UrlGetLocation( psz1 : PChar ) : PChar
6372: Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD ) : HRESULT
6373: Function UrlEscape(pszUrl : PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD ) : HRESULT
6374: Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD ) : HRESULT
6375: Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD ) : HRESULT
6376: Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6377: Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD ) : HRESULT
6378: Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD ) : HRESULT
6379: Function HashData( pData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6380: Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD ) : HRESULT
6381: Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD ) : HRESULT
6382: Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6383: Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6384: Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar ) : DWORD
6385: Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD ) : Longint
6386: Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
pcchValueName:DWORD;pdwType:DWORD; pvData : __Pointer; pcbData : DWORD ) : Longint
6387: Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcch.MaxValueNameLen:DWORD):Longint;
6388: Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD ) : DWORD
6389: Function SHRegGetPath(hKey:HKEY; pcSzSubKey,pcSzValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6390: Function SHRegSetPath( hKey:HKEY; pcSzSubKey, pcSzValue, pcSzPath : PChar; dwFlags : DWORD): DWORD
6391: SHREGDEL_DEFAULT', 'LongWord( $00000000 );
6392: SHREGDEL_HKCU', 'LongWord( $00000001 );
6393: SHREGDEL_HKLM', 'LongWord( $00000010 );
6394: SHREGDEL_BOTH', 'LongWord( $00000011 );
6395: SHREGENUM_DEFAULT', 'LongWord( $00000000 );
6396: SHREGENUM_HKCU', 'LongWord( $00000001 );
6397: SHREGENUM_HKLM', 'LongWord( $00000010 );
6398: SHREGENUM_BOTH', 'LongWord( $00000011 );
6399: SHREGSET_HKCU', 'LongWord( $00000001 );
6400: SHREGSET_FORCE_HKCU', 'LongWord( $00000002 );
6401: SHREGSET_HKLM', 'LongWord( $00000004 );
6402: SHREGSET_FORCE_HKLM', 'LongWord( $00000008 );
6403: TSHRegDelFlags', 'DWORD
6404: TSHRegEnumFlags', 'DWORD
6405: HUSKEY', 'THandle
6406: ASSOCF_INIT_NOREMAPCLSID', 'LongWord( $00000001 );
6407: ASSOCF_INIT_BYEXENAME', 'LongWord( $00000002 );
6408: ASSOCF_OPEN_BYEXENAME', 'LongWord( $00000002 );
6409: ASSOCF_INIT_DEFAULTTOSTAR', 'LongWord( $00000004 );
6410: ASSOCF_INIT_DEFAULTTOFOLDER', 'LongWord( $00000008 );
6411: ASSOCF_NOUSERSETTINGS', 'LongWord( $00000010 );
6412: ASSOCF_NOTRUNCATE', 'LongWord( $00000020 );
6413: ASSOCF_VERIFY', 'LongWord( $00000040 );
6414: ASSOCF_REMAPPRUNDLL', 'LongWord( $00000080 );
6415: ASSOCF_NOFIXUPS', 'LongWord( $00000100 );
6416: ASSOCF_IGNOREBASECLASS', 'LongWord( $00000200 );
6417: ASSOCF', 'DWORD
6418: ASSOCSTR_COMMAND', 'LongInt'( 1 );
6419: ASSOCSTR_EXECUTABLE', 'LongInt'( 2 );
6420: ASSOCSTR_FRIENDLYDOCNAME', 'LongInt'( 3 );
6421: ASSOCSTR_FRIENDLYAPPNAME', 'LongInt'( 4 );
6422: ASSOCSTR_NOOPEN', 'LongInt'( 5 );
6423: ASSOCSTR_SHELLNEWVALUE', 'LongInt'( 6 );
6424: ASSOCSTR_DDECOMMAND', 'LongInt'( 7 );
6425: ASSOCSTR_DDEIFEXEC', 'LongInt'( 8 );
6426: ASSOCSTR_DDEAPPLICATION', 'LongInt'( 9 );

```

```

6427: ASSOCSTR_DDETOPIC', 'LongInt'( 10);
6428: ASSOCSTR_INFOTIP', 'LongInt'( 11);
6429: ASSOCSTR_MAX', 'LongInt'( 12);
6430: ASSOCSTR', 'DWORD
6431: ASSOCKEY_SHELLEXECCLASS', 'LongInt'( 1);
6432: ASSOCKEY_APP', 'LongInt'( 2);
6433: ASSOCKEY_CLASS', 'LongInt'( 3);
6434: ASSOCKEY_BASECLASS', 'LongInt'( 4);
6435: ASSOCKEY_MAX', 'LongInt'( 5);
6436: ASSOCKEY', 'DWORD
6437: ASSOCDATA_MSIDESCRIPTOR', 'LongInt'( 1);
6438: ASSOCDATA_NOACTIVATEHANDLER', 'LongInt'( 2);
6439: ASSOCDATA_QUERYCLASSTORE', 'LongInt'( 3);
6440: ASSOCDATA_HASPERUSERASSOC', 'LongInt'( 4);
6441: ASSOCDATA_MAX', 'LongInt'( 5);
6442: ASSOCDATA', 'DWORD
6443: ASSOCENUM_NONE', 'LongInt'( 0);
6444: ASSOCENUM', 'DWORD
6445: SID_IQueryAssociations', 'String' '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6446: SHACF_DEFAULT $00000000;
6447: SHACF_FILESYSTEM', 'LongWord( $00000001);
6448: SHACF_URLHISTORY', 'LongWord( $00000002);
6449: SHACF_URLMRU', 'LongWord( $00000004);
6450: SHACF_USETAB', 'LongWord( $00000008);
6451: SHACF_FILESYS_ONLY', 'LongWord( $00000010);
6452: SHACF_AUTOSUGGEST_FORCE_ON', 'LongWord( $10000000);
6453: SHACF_AUTOSUGGEST_FORCE_OFF', 'LongWord( $20000000);
6454: SHACF_AUTOAPPEND_FORCE_ON', 'LongWord( $40000000);
6455: SHACF_AUTOAPPEND_FORCE_OFF', 'LongWord( DWORD( $80000000 ) );
6456: Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD ) : HRESULT
6457: Procedure SHSetThreadRef( punk : IUnknown )
6458: Procedure SHGetThreadRef( out ppunk : IUnknown )
6459: CTF_INSIST', 'LongWord( $00000001);
6460: CTF_THREAD_REF', 'LongWord( $00000002);
6461: CTF_PROCESS_REF', 'LongWord( $00000004);
6462: CTF_COINIT', 'LongWord( $00000008);
6463: Function SHCreateShellPalette( hdc : HDC ) : HPALETTE
6464: Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD )
6465: Function ColorHLSToRGB( whue, wluminance, wsaturation : WORD ) : TColorRef
6466: Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean ) : TColorRef
6467: Function GetSysColorBrush( nIndex : Integer ) : HBRUSH
6468: Function DrawFocusRect( hdc : HDC; const lprc : TRect ) : BOOL
6469: Function FillRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH ) : Integer
6470: Function FrameRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH ) : Integer
6471: Function InvertRect( hdc : HDC; const lprc : TRect ) : BOOL
6472: Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer ) : BOOL
6473: Function SetRectEmpty( var lprc : TRect ) : BOOL
6474: Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect ) : BOOL
6475: Function InflateRect( var lprc : TRect; dx, dy : Integer ) : BOOL
6476: Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect ) : BOOL
6477: Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect ) : BOOL
6478:
6479: Function InitializeFlatSB( hWnd : HWND ) : Bool
6480: Procedure UninitializeFlatSB( hWnd : HWND )
6481: Function FlatSB_GetScrollProp( pl : HWND; propIndex : Integer; p3 : PInteger ) : Bool
6482: Function FlatSB_SetScrollProp( pl : HWND; index : Integer; newValue : Integer; p4 : Bool ) : Bool
6483: Function GET_APPCOMMAND_LPARAM( lParam : Integer ) : Word //of JwWin32
6484: Function GET_DEVICE_LPARAM( lParam : Integer ) : Word
6485: Function GET_MOUSEKEY_LPARAM( lParam : Integer ) : Integer
6486: Function GET_FLAGS_LPARAM( lParam : Integer ) : Word
6487: Function GET_KEYSTATE_LPARAM( lParam : Integer ) : Word
6488:
6489:
6490: // **** 204 unit uPSI_ShellAPI;
6491: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT ) : UINT
6492: Function DragQueryPoint( Drop : HDROP; var Point : TPoint ) : BOOL
6493: Procedure DragFinish( Drop : HDROP )
6494: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL )
6495: Function ShellExecute( hWnd : HWND; Operation, FileName, Parameters, Directory : PChar; ShowCmd : Integer ) : HINST
6496: Function FindExecutable( FileName, Directory : PChar; Result : PChar ) : HINST
6497: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON ) : Integer
6498: Function DuplicateIcon( hInst : HINST; Icon : HICON ) : HICON
6499: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word ) : HICON
6500: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT ) : HICON
6501: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData ) : UINT
6502: Function DoEnvironmentSubst( szString : PChar; cbString : UINT ) : DWORD
6503: Function ExtractIconEx( lpszFile : PChar; nIconIndex : Int; var phiconLarge, phiconSmall : HICON; nIcons : UINT ) : UINT
6504: Procedure SHFreeNameMappings( hNameMappings : THandle )
6505:
6506: DLLVER_PLATFORM_WINDOWS', 'LongWord( $00000001);
6507: DLLVER_PLATFORM_NT', 'LongWord( $00000002);
6508: DLLVER_MAJOR_MASK', 'LongWord( Int64( $FFFF000000000000 ) );
6509: DLLVER_MINOR_MASK', 'LongWord( Int64( $0000FFFF00000000 ) );
6510: DLLVER_BUILD_MASK', 'LongWord( Int64( $00000000FFFF0000 ) );
6511: DLLVER_QFE_MASK', 'LongWord( Int64( $000000000000FFFF ) );
6512: Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word ) : Int64
6513: Function SimpleXMLEncode( const S : string ) : string
6514: Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean )
6515: Function XMLEncode( const S : string ) : string

```

```

6516: Function XMLDecode( const S : string) : string
6517: Function EntityEncode( const S : string) : string
6518: Function EntityDecode( const S : string) : string
6519:
6520: procedure RIRegister_CPort_Routines(S: TPSEExec);
6521: Procedure EnumComPorts( Ports : TStrings)
6522: Procedure ListComPorts( Ports : TStrings)
6523: Procedure ComPorts( Ports : TStrings) //Alias to Arduino
6524: Function GetComPorts: TStringlist;
6525: Function StrToBaudRate( Str : string) : TBaudRate
6526: Function StrToStopBits( Str : string) : TStopBits
6527: Function StrToDataBits( Str : string) : TDataBits
6528: Function StrToParity( Str : string) : TParityBits
6529: Function StrToFlowControl( Str : string) : TFlowControl
6530: Function BaudRateToStr( BaudRate : TBaudRate) : string
6531: Function StopBitsToStr( StopBits : TStopBits) : string
6532: Function DataBitsToStr( DataBits : TDataBits) : string
6533: Function ParityToStr( Parity : TParityBits) : string
6534: Function FlowControlToStr( FlowControl : TFlowControl) : string
6535: Function ComErrorsToStr( Errors : TComErrors) : string
6536:
6537: Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6538: Function DispatchMessage( const lpMsg : TMsg) : Longint
6539: Function TranslateMessage( const lpMsg : TMsg) : BOOL
6540: Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6541: Function PeekMessage( var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6542: Function GetMessagePos : DWORD
6543: Function GetMessageTime : Longint
6544: Function GetMessageExtraInfo : Longint
6545: Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6546: Procedure JAddToRecentDocs( const Filename : string)
6547: Procedure ClearRecentDocs
6548: Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6549: Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6550: Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6551: Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6552: Function RecycleFile( FileToRecycle : string) : Boolean
6553: Function JCopyFile( FromFile, ToDir : string) : Boolean
6554: Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6555: Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6556: Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL
6557: Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL
6558: Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL
6559: Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD; dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD;pszGroupName: LP
6560: Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD; dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD; pszGroupNameName
6561: Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD; dwServiceState : DWORD;lpServices : LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD; pszGroupNameName
6562: Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6563:
6564: ***** unit uPSI_JclPeImage;
6565:
6566: Function IsValidPeFile( const FileName : TFileName) : Boolean
6567: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6568: Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6569: Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize : DWORD) : TJclRebaseImageInfo
6570: Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6571: Function PeClearCheckSum( const FileName : TFileName) : Boolean
6572: Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6573: Function PeDoesExportFunction(const FileName:TFileName;const FuncName:string;Options:TJclSmartCompOptions):Bool;
6574: Function PeIsExportFunctionForwardedEx( const FileName; const FunctionName : string; var ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6575: Function PeIsExportFunctionForwarded(const FileName:TFileName;const FunctionName:string;Options:TJclSmartCompOptions):Bool
6576: Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName : string; Options : TJclSmartCompOptions) : Boolean
6577: Function PeDoesImportLibrary(const FileName:TFileName;const LibraryName:string;Recursive:Boolean):Boolean;
6578: Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive : Boolean; FullPathName : Boolean) : Boolean
6579: Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const LibraryName:string; IncludeLibNames : Boolean): Boolean
6580: Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6581: Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6582: Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6583: Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const NamesList:TStrings):Bool
6584: Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6585: Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName, Descript:Bool):Bool;

```

```

6586: Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings ) : Boolean;
6587: Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings ) : Boolean;
6588: Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6589: //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6590: //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6591: //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6592: //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string ) :
   PImageSectionHeader
6593: //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6594: //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6595: Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):_
   Pointer;
6596:   SIRegister_TJclPeSectionStream(CL);
6597:   SIRegister_TJclPeMapImgHookItem(CL);
6598:   SIRegister_TJclPeMapImgHooks(CL);
6599: //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
   NtHeaders:TImageNtHeaders):Boolean
6600: //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6601: Type TJclBorUmSymbolKind,'(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6602: TJclBorUmSymbolModifier,'( smQualified, smLinkProc )
6603: TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6604: TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6605: TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6606: TJclPeUmResult', '( unNotMangled, umBorland, umMicrosoft )
6607: Function PeBorUmangleName( const Name : string; var Unmangled : string; var Description :
   TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6608: Function PeBorUmangleName1(const Name:string;var Unmangled:string;var
   Description:TJclBorUmDescription):TJclBorUmResult;
6609: Function PeBorUmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6610: Function PeBorUmangleName3( const Name : string) : string;
6611: Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6612: Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6613:
6614:
6615: //***** SysTools uPSI_StSystem; *****
6616: Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6617: Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6618: Function DeleteVolumeLabel( Drive : Char) : Cardinal
6619: //Procedure EnumerateDirectories(const
   StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6620: //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
   IncludeItem:TIncludeItemFunc);
6621: Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6622: Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6623: Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6624: Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6625: Function FlushOsBuffers( Handle : Integer) : Boolean
6626: Function GetCurrentUser : AnsiString
6627: Function GetDiskClass( Drive : Char) : DiskClass
6628: Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
   SectorsPerCluster:Cardinal):Bool;
6629: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
   DiskSize:Double):Bool;
6630: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
   DiskSize:Comp):Boolean;
6631: { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes }
6632: Function getDiskSpace2(const path: String; index: integer): int64;
6633: Function GetFileCreateDate( const FileName : Ansistring) : TDateTime
6634: Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6635: Function GetfileLastModify( const FileName : Ansistring) : TDateTime
6636: Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6637: Function GetLongPath( const APath : AnsiString) : AnsiString
6638: Function GetMachineName : AnsiString
6639: Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6640: Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6641: Function GetShortPath( const APath : AnsiString) : AnsiString
6642: Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6643: Function GetTempFolder( aForceSlash : boolean) : AnsiString
6644: Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6645: Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6646: Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6647: Function StIsDirectory( const DirName : AnsiString) : Boolean
6648: Function IsDirectoryEmpty( const S : AnsiString) : Integer
6649: Function IsDriveReady( Drive : Char) : Boolean
6650: Function IsFile( const FileName : AnsiString) : Boolean
6651: Function IsFileArchive( const S : AnsiString) : Integer
6652: Function IsFileHidden( const S : AnsiString) : Integer
6653: Function IsFileReadOnly( const S : AnsiString) : Integer
6654: Function IsFileSystem( const S : AnsiString) : Integer
6655: Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6656: Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6657: Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6658: Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6659: Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6660: Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6661: Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6662: Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6663: Function ValidDrive( Drive : Char) : Boolean
6664: Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal

```

```

6665:
6666: //*****unit uPSI_JclLANMan;*****
6667: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
6668:   const PasswordNeverExpires : Boolean ) : Boolean
6669:   Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
6670:     const PasswordNeverExpires : Boolean ) : Boolean
6671:   Function DeleteAccount( const Servername, Username : string ) : Boolean
6672:   Function DeleteLocalAccount( Username : string ) : Boolean
6673:   Function CreateGlobalGroup( const Server, Groupname, Description : string ) : Boolean
6674:   Function DeleteLocalGroup( const Server, Groupname : string ) : Boolean
6675:   Function GetLocalGroups( const Server : string; const Groups : TStrings ) : Boolean
6676:   Function GetGlobalGroups( const Server : string; const Groups : TStrings ) : Boolean
6677:   Function LocalGroupExists( const Group : string ) : Boolean
6678:   Function GlobalGroupExists( const Server, Group : string ) : Boolean
6679:   Function AddAccountToLocalGroup( const Accountname, Groupname : string ) : Boolean
6680:   Procedure LookupGroupName( const Server : string; const RID : TNetWellKnownRID ) : string
6681:   Function ParseAccountName( const QualifiedName : string; var Domain, UserName : string )
6682:   Function IsLocalAccount( const AccountName : string ) : Boolean
6683:   Function TimeStampInterval( StartStamp, EndStamp : TDateTime ) : integer
6684:   Function GetRandomString( NumChar : cardinal ) : string
6685: //*****unit uPSI_cUtils;*****
6686: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )'
6687: Function cIsWinNT : boolean
6688: Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
6689: Multitasking:Boolean;
6690:   Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ) : THandle
6691:   Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
6692: CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean ) : string
6693:   Function cGetShortName( FileName : string ) : string
6694:   Procedure cShowError( Msg : String )
6695:   Function cCommaStrToStr( s : string; formatstr : string ) : string
6696:   Function cIncludeQuoteIfSpaces( s : string ) : string
6697:   Function cIncludeQuoteIfNeeded( s : string ) : string
6698:   Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream )
6699:   Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6700:   Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET ) : boolean;
6701:   Function cBuildFilter1( var value : string; const _filters : array of string ) : boolean;
6702:   Function cCodeInstoStr( s : string ) : string
6703:   Function cStrtoCodeIns( s : string ) : string
6704:   Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6705:   Function cStrtoPoint( var pt : TPoint; value : string )
6706:   Function cPointtoStr( const pt : TPoint ) : string
6707:   Function cListtoStr( const List : TStrings ) : string
6708:   Function ListtoStr( const List : TStrings ) : string
6709:   Procedure StrtoList( s : string; const List : TStrings; const delimiter : char )
6710:   Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char )
6711:   Function cGetFileType( const FileName : string ) : TUnitType
6712:   Function cGetExTyp( const FileName : string ) : TExUnitType
6713:   Procedure cSetPath( Add : string; const UseOriginal : boolean )
6714:   Function cExpandFileto( const FileName : string; const basePath : string ) : string
6715:   Function cFileSamePath( const FileName : string; const TestPath : string ) : boolean
6716:   Procedure cCloneMenu( const FromMenuItem : TMenuItem; ToMenuItem : TMenuItem )
6717:   Function cGetLastPos( const SubStr : string; const S : string ) : integer
6718:   Function cGenMakePath( FileName : String ) : String;
6719:   Function cGenMakePath2( FileName : String ) : String
6720:   Function cGetRealPath( BrokenFileName : String; Directory : String ) : String
6721:   Function cCalcMod( Count : Integer ) : Integer
6722:   Function cGetVersionString( FileName : string ) : string
6723:   Function cCheckChangeDir( var Dir : string ) : boolean
6724:   Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6725:   Function cIsNumeric( s : string ) : boolean
6726:   Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6727:   Function AttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6728:   Function GetfileTyp( const FileName : string ) : TUnitType
6729:   Function Atoi(const aStr: string): integer
6730:   Function Itoa(const aint: integer): string
6731:   Function Atof(const aStr: string): double';
6732:   Function Atol(const aStr: string): longint';
6733:
6734:
6735: procedure SIRegister_cHTTPUtils(CL: TPPascalCompiler);
6736: begin
6737:   FindClass( 'TOBJECT' ),'EHTTP
6738:   FindClass( 'TOBJECT' ),'EHTTPParser
6739:   //AnsiCharSet', 'set of AnsiChar
6740:   AnsiStringArray', 'array of AnsiString
6741:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6742:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6743:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6744:   +'TPPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6745:   +'CustomMinVersion : Integer; end
6746:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6747:   +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6748:   +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6749:   +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'

```

```

6750: +'ooke, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6751: +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6752: +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges, '
6753: +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSince'
6754: +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6755: +'nection, hntOrigin, hntKeepAlive )
6756: THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6757: THTTPCustomHeader', 'record FieldName : AnsiString;FieldValue :'
6758: +' AnsiString; end
6759: //THTTPCustomHeader', '^THTTPCustomHeader // will not work
6760: THTTPContentLengthEnum', '( hcLNone, hcLByteCount )
6761: THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6762: //THTTPContentLength', '^THTTPContentLength // will not work
6763: THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6764: THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6765: +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6766: +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6767: +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScipt, hctAppli'
6768: +'ctionCustom, hctAudioCustom, hctVideoCustom )
6769: THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6770: +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray; '
6771: +'CustomStr : AnsiString; end
6772: THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )'
6773: THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6774: +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6775: +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6776: +'String; DateTime : TDateTime; Custom : AnsiString; end
6777: THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )'
6778: THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnum'
6779: +'m; Custom : AnsiString; end
6780: THTTPConnectionFieldEnum', '( hcfnNone, hcfcCustom, hcfcClose, hcfcKeepAlive )'
6781: THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum; '
6782: +' Custom : AnsiString; end
6783: THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )'
6784: THTTPAgeField', 'record Value : THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6785: THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )'
6786: THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6787: +' , hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )'
6788: THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6789: +' , hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6790: +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )'
6791: THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6792: THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6793: +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )'
6794: THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6795: THTTPContentEncodingException', '( hcfeNone, hcfeList )'
6796: THTTPContentEncodingException', 'record Value : THTTPContentEncoding'
6797: +'FieldEnum; List : array of THTTPContentEncoding; end
6798: THTTPRetryAfterFieldEnum', '( hrarNone, hrarCustom, harfDate, harfSeconds )'
6799: THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum; '
6800: +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6801: THTTPContentRangeFieldEnum', '( hrcfNone, hrcfCustom, hrcfByteRange )'
6802: THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6803: +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6804: THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )'
6805: THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6806: THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6807: THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6808: +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6809: +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6810: +'CustomFieldArray; Custom : AnsiString; end
6811: //THTTPSetCookieField', '^THTTPSetCookieField // will not work
6812: THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6813: THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )'
6814: THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6815: //THTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6816: THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6817: THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6818: +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6819: THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6820: +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength; '
6821: +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6822: +' ; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6823: THTTPCustomHeaders', 'array of THTTPCustomHeader
6824: //THTTPFixedHeaders', 'array[THTTPHeaderNameEnum] of AnsiString
6825: THTTPFixedHeaders', 'array[0..42] of AnsiString
6826: THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6827: +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )'
6828: THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6829: THTTPRequestStartLine', 'record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6830: THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders; '
6831: +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6832: +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6833: //THTTPRequestHeader', '^THTTPRequestHeader // will not work
6834: THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6835: +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6836: THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)'
6837: THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6838: +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end

```

```

6839: THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6840:   +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6841:   +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6842:   +' THTTPDateField; Age : THTTPAgeField; end
6843: //THTTPResponseHeader', '^THTTPResponseHeader // will not work
6844: THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6845:   +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6846: Function HTTPMessageHasContent( const H : THTTPCommonHeaders ) : Boolean
6847: Procedure InitHTTPRequest( var A : THTTPRequest )
6848: Procedure InitHTTPResponse( var A : THTTPResponse )
6849: Procedure ClearHTTPVersion( var A : THTTPVersion )
6850: Procedure ClearHTTPContentLength( var A : THTTPContentLength )
6851: Procedure ClearHTTPContentType( var A : THTTPContentType )
6852: Procedure ClearHTTPDateField( var A : THTTPDateField )
6853: Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding )
6854: Procedure ClearHTTPConnectionField( var A : THTTPConnectionField )
6855: Procedure ClearHTTPAgeField( var A : THTTPAgeField )
6856: Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding )
6857: Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField )
6858: Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField )
6859: Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField )
6860: Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders )
6861: //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders )
6862: Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders )
6863: Procedure ClearHTTPCookieField( var A : THTTPCookieField )
6864: Procedure ClearHTTPMethod( var A : THTTPMethod )
6865: Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine )
6866: Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader )
6867: Procedure ClearHTTPRequest( var A : THTTPRequest )
6868: Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine )
6869: Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader )
6870: Procedure ClearHTTPResponse( var A : THTTPResponse )
6871: THTTPStringOption', '( hsoNone )
6872: THTTPStringOptions', 'set of THTTPStringOption
6873: FindClass('TOBJECT'), 'TansiStringBuilder
6874:
6875: Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TansiStringBuilder; P:THTTPStringOptions;
6876: Procedure BuildStrHTTPContentLengthValue(const
6877: A:THTTPContentLength;B:TansiStringBuilder;P:THTTPStringOptions)
6878: Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
6879: B:TansiStringBuilder;P:THTTPStringOptions)
6880: Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TansiStringBuilder;const
6881: P:THTTPStringOptions)
6882: Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TansiStringBuilder; const
6883: P:THTTPStringOptions)
6884: Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
6885: B : TansiStringBuilder; const P : THTTPStringOptions)
6886: Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TansiStringBuilder; const P :
6887: THTTPStringOptions)
6888: Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TansiStringBuilder;const
6889: P:THTTPStringOptions);
6890: Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
6891: TansiStringBuilder; const P : THTTPStringOptions)
6892: Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TansiStringBuilder;
6893: const P : THTTPStringOptions)
6894: Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TansiStringBuilder;
6895: const P : THTTPStringOptions)
6896: Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TansiStringBuilder;
6897: const P : THTTPStringOptions)
6898: Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TansiStringBuilder;
6899: const P : THTTPStringOptions)
6900: Procedure BuildStrHTTPProxyConnectionField( const A : THTTPProxyConnectionField; const B : TansiStringBuilder;
6901: const P : THTTPStringOptions);

```

```

6902: Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B : 
TAnsiStringBuilder; const P : THTTPStringOptions)
6903: Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P 
THTTPStrOptions);
6904: Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const P:THTTPStringOptions);
6905: Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const P:THTTPStringOptions);
6906: Function HTTPContentTypeValueToStr( const A : THTTPContentType ) : AnsiString
6907: Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField ) : AnsiString
6908: Function HTTPCookieFieldValueToStr( const A : THTTPCookieField ) : AnsiString
6909: Function HTTPMethodToStr( const A : THTTPMethod ) : AnsiString
6910: Function HTTPRequestToStr( const A : THTTPRequest ) : AnsiString
6911: Function HTTPResponseToStr( const A : THTTPResponse ) : AnsiString
6912: Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const 
Domain:AnsiString;const Secure:Boolean; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT' 
+PHeaderNameEnum; const HeaderPtr : __Pointer ) : Boolean
6913: SIRegister_THTTPParser(CL);
6914: FindClass('TOBJECT','THTTPContentDecoder'
6915: THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder )
6916: THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6917: THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,
6918: +' crcsContentCRLF, crcsTrailer, crcsFinished )
6919: THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String )
6920: SIRegister_THTTPContentDecoder(CL);
6921: THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6922: FindClass('TOBJECT','THTTPContentReader'
6923: THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader )
6924: THTTPContentReaderLogEvent', 'Procedure ( const Sender : THTTPContentReader; const LogMsg:String; const 
LogLevel:Int ;
6925: SIRegister_THTTPContentReader(CL);
6926: THTTPContentWriterMechanism', '( hctmEvent, hctmString, hctmStream, hctmFile )
6927: FindClass('TOBJECT','THTTPContentWriter'
6928: THTTPContentWriterLogEvent', 'Procedure ( const Sender : THTTPContentWriter; const LogMsg:AnsiString );
6929: SIRegister_THTTPContentWriter(CL);
6930: Procedure SelfTestcHTTPUtils
6931: end;
6932:
6933:
6934: (*-----*)
6935: procedure SIRegister_cTLSUtils(CL: TPPascalCompiler);
6936: begin
6937: 'TLSLibraryVersion', 'String '1.00
6938: 'TLSerror_None', 'LongInt'( 0 );
6939: 'TLSerror_InvalidBuffer', 'LongInt'( 1 );
6940: 'TLSerror_InvalidParameter', 'LongInt'( 2 );
6941: 'TLSerror_InvalidCertificate', 'LongInt'( 3 );
6942: 'TLSerror_InvalidState', 'LongInt'( 4 );
6943: 'TLSerror_DecodeError', 'LongInt'( 5 );
6944: 'TLSerror_BadProtocol', 'LongInt'( 6 );
6945: Function TLSErrorMessage( const TLSerror : Integer ) : String
6946: SIRegister_ETLSerror(CL);
6947: TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6948: TTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6949: Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion )
6950: Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion )
6951: Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion )
6952: Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion )
6953: Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion ) : Boolean
6954: Function IsSSL2( const A : TTLSProtocolVersion ) : Boolean
6955: Function IsSSL3( const A : TTLSProtocolVersion ) : Boolean
6956: Function IsTLS10( const A : TTLSProtocolVersion ) : Boolean
6957: Function IsTLS11( const A : TTLSProtocolVersion ) : Boolean
6958: Function IsTLS12( const A : TTLSProtocolVersion ) : Boolean
6959: Function IsTLS10OrLater( const A : TTLSProtocolVersion ) : Boolean
6960: Function IsTLS11OrLater( const A : TTLSProtocolVersion ) : Boolean
6961: Function IsTLS12OrLater( const A : TTLSProtocolVersion ) : Boolean
6962: Function IsFutureTLSVersion( const A : TTLSProtocolVersion ) : Boolean
6963: Function IsKnownTLSVersion( const A : TTLSProtocolVersion ) : Boolean
6964: Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion ) : String
6965: Function TLSProtocolVersionName( const A : TTLSProtocolVersion ) : String
6966: PTLSRandom', '^TTLSRandom // will not work
6967: Procedure InitTLSRandom( var Random : TTLSRandom )
6968: Function TLSRandomToStr( const Random : TTLSRandom ) : AnsiString
6969: 'TLSSessionIDMaxLen', 'LongInt'( 32 );
6970: Procedure InitTLSSessionID( var SessionID : TLSSessionID; const A : AnsiString )
6971: Function EncodedTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TLSSessionID):Int;
6972: Function DecodedTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TLSSessionID):Int;
6973: TTLSignatureAndHashAlgorithm', 'record Hash : TTLSShHashAlgorithm'
6974: +' ; Signature : TTLSignatureAlgorithm; end
6975: // TTLSignatureAndHashAlgorithm', '^TTLSignatureAndHashAlgorithm +'// will not work
6976: TTLSignatureAndHashAlgorithmArray', 'array of TTLSignatureAndHashAlgorithm
6977: TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6978: +' DSS, tlskeaDH_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6979: TTLSMACAlgorithm', '( tlsmacNone, tlsmacNULL, tlsmacHMAC_MD5, tlsmac'
6980: +' HMAC_SHA1, tlsmacHMAC_SHA256, tlsmacHMAC_SHA384, tlsmacHMAC_SHA512 )
6981: TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6982: +' nteger; Supported : Boolean; end
6983: PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6984: 'TLS_MAC_MAXDIGESTSIZE', 'LongInt'( 64 );

```

```

6985: TTLSPRFAlgorithm', '( tlspaSHA256 )
6986: Function tlsp_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6987: Function tlsp_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6988: Function tlsp_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6989: Function tlsp_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6990: Function tlsp10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6991: Function tlsp12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6992: Function tlsp12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6993: Function TLSPRF( const ProtoVersion:TTLSProtocolVersion, const Secret, ALabel, Seed: AString; const
Size:Int):AString;
6994: Function tlsl0KeyBlock(const MasterSecret, ServerRandom, ClientRandom:AnsiString; const
Size:Integer):AnsiString
6995: Function tlsl2SHA256KeyBlock(const MasterSecret, ServerRandom, ClientRandom: AnsiString; const
Size:Int):AnsiString;
6996: Function tlsl2SHA512KeyBlock(const MasterSecret, ServerRandom, ClientRandom: AnsiString; const
Size:Int):AnsiString;
6997: Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
ClientRandom : AnsiString; const Size : Integer) : AnsiString
6998: Function tlsl0MasterSecret(const PreMasterSecret, ClientRandom, ServerRandom:AnsiString) :AnsiString;
6999: Function tlsl2SHA256MasterSecret(const PreMasterSecret, ClientRandom, ServerRandom:AnsiString):AnsiString;
7000: Function tlsl2SHA512MasterSecret(const PreMasterSecret, ClientRandom, ServerRandom:AnsiString): AnsiString;
7001: Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret, ClientRandom,
ServerRandom:AnsiString) : AnsiString
7002: TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
7003: +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
7004: +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
7005: Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
7006: Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
7007: 'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
7008: 'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
7009: Procedure SelfTestcTLSUtils
7010: end;
7011:
7012: (*-----*)
7013: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
7014: begin
7015:   sPosData', 'record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
integer; disks : integer; mx : integer; my : integer; end
7016: // pBoard', '^tBoard // will not work
7017: Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
7018: Function rCheckMove( color : byte; cx, cy : integer) : integer
7019: //Function rDoStep( data : pBoard) : word
7020: Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
7021: end;
7022:
7023: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
7024: begin
7025: Function InEditMode( ADataset : TDataset) : Boolean
7026: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
7027: Function CheckDataSource1( ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
7028: Function GetFieldText( AField : TField) : String
7029: end;
7030:
7031: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
7032: begin
7033:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7034: TMyPrintRange', '( prAll, prSelected )
7035: TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7036: +'ded, ssDateTime, ssTime, ssCustom )
7037: TSortDirection', '( sdAscending, sdDescending )
7038: TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7039: TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
7040: +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7041: TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7042: TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7043: SIRegister_TSortOptions(CL);
7044: SIRegister_TPrintOptions(CL);
7045: TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7046: SIRegister_TSortedList(CL);
7047: TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7048: TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7049: TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7050: TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7051: +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7052: SIRegister_TFontSetting(CL);
7053: SIRegister_TFontlist(CL);
7054: AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row : '
7055: +' integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7056: TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7057: TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7058: TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
7059: TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7060: TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7061: TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7062: TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : integer'
7063: +'r; var SortStyle : TSortStyle)
7064: TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow : '

```

```

7065: +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7066: SIRegister_TSortGrid(CL);
7067: Function ExtendedCompare( const Str1, Str2 : String) : Integer
7068: Function NormalCompare( const Str1, Str2 : String) : Integer
7069: Function DateTimeCompare( const Str1, Str2 : String) : Integer
7070: Function NumericCompare( const Str1, Str2 : String) : Integer
7071: Function TimeCompare( const Str1, Str2 : String) : Integer
7072: //Function Compare( Item1, Item2 : Pointer) : Integer
7073: end;
7074:
7075: ***** procedure Register_IB(CL: TPSPascalCompiler);
7076: Procedure IBAalloc( var P, OldSize, NewSize : Integer)
7077: Procedure IBEerror( ErrMess : TIBClientError; const Args : array of const)
7078: Procedure IB DataBaseError
7079: Function StatusVector : PISC_STATUS
7080: Function StatusVectorArray : PStatusVector
7081: Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
7082: Function StatusVectorAsText : string
7083: Procedure SetIB DataBaseErrorMessages( Value : TIB DataBaseErrorMessages)
7084: Function GetIB DataBaseErrorMessages : TIB DataBaseErrorMessages
7085:
7086:
7087: //*****unit uPSI_BoldUtils;*****
7088: Function CharCount( c : char; const s : string) : integer
7089: Function BoldNamesEqual( const name1, name2 : string) : Boolean
7090: Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7091: Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7092: Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7093: Function BoldTrim( const S : string) : string
7094: Function BoldIsPrefix( const S, Prefix : string) : Boolean
7095: Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7096: Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7097: Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7098: Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7099: Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7100: //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7101: Function CapitalisedToSpaced( Capitalised : String) : String
7102: Function SpacedToCapitalised( Spaced : String) : String
7103: Function BooleanToString( BoolValue : Boolean) : String
7104: Function StringToBoolean( StrValue : String) : Boolean
7105: Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7106: Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7107: Function StringListToVarArray( List : TStringList) : variant
7108: Function IsLocalMachine( const Machinename : WideString) : Boolean
7109: Function GetComputerNameStr : string
7110: Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7111: Function BoldStrToDateFmt( const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7112: Function BoldDateToStrFmt( const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7113: Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7114: Function BoldParseFormattedDate(const value:String;const formats:array of string; var Date:TDateTime):Boolean;
    Date:TDateTime;
7115: Procedure EnsureTrailing( var Str : String; ch : char)
7116: Function BoldDirectoryExists( const Name : string) : Boolean
7117: Function BoldForceDirectories( Dir : string) : Boolean
7118: Function BoldRootRegistryKey : string
7119: Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7120: Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7121: Function LogicalAnd( A, B : Integer) : Boolean
7122: record TByHandleFileInformation dwFileAttributes : DWORD;
7123:   +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7124:   +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSize'
7125:   +'nLow : DWORD; nNumberOfLinks : DWORD; nfileIndexHigh : DWORD; nfileIndexLow : DWORD; end
7126: Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7127: Function IsFirstInstance : Boolean
7128: Procedure ActivateFirst( AString : PChar)
7129: Procedure ActivateFirstCommandLine
7130: function MakeAckPkt(const BlockNumber: Word): string;
7131: procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string);
7132: procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7133: procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer; E: Exception); overload;
7134: procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7135: function IdStrToWord(const Value: String): Word;
7136: function IdWordToStr(const Value: Word): WordStr;
7137: Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7138: Function CPUFeatures : TCPUFeatures
7139:
7140: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7141: begin
7142:   AddTypeS('TXRTLBitIndex', 'Integer'
7143:   Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7144:   Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7145:   Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7146:   Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7147:   Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7148:   Function XRTLSwapHiLo16( X : Word) : Word
7149:   Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7150:   Function XRTLSwapHiLo64( X : Int64) : Int64
7151:   Function XRTLROL32( A, S : Cardinal) : Cardinal
7152:   Function XRTLROL32( A, S : Cardinal) : Cardinal

```

```

7153: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7154: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7155: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7156: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7157: //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7158: //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7159: Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer )
7160: Function XRTLPopulation( A : Cardinal ) : Cardinal
7161: end;
7162:
7163: Function XRTLURLDecode( const ASrc : WideString ) : WideString
7164: Function XRTLURLEncode( const ASrc : WideString ) : WideString
7165: Function XRTLURINormalize( const AURI : WideString ) : WideString
7166: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
    VPassword : WideString)
7167: Function XRTLExtractLongPathName(APath: string): string;
7168:
7169: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7170: begin
7171:   AddTypeS('Int8', 'ShortInt'
7172:   AddTypeS('Int16', 'SmallInt'
7173:   AddTypeS('Int32', 'LongInt'
7174:   AddTypeS('UInt8', 'Byte'
7175:   AddTypeS('UInt16', 'Word'
7176:   AddTypeS('UInt32', 'LongWord'
7177:   AddTypeS('UInt64', 'Int64'
7178:   AddTypeS('Word8', 'UInt8'
7179:   AddTypeS('Word16', 'UInt16'
7180:   AddTypeS('Word32', 'UInt32'
7181:   AddTypeS('Word64', 'UInt64'
7182:   AddTypeS('LargeInt', 'Int64'
7183:   AddTypeS('NativeInt', 'Integer'
7184:   AddTypeS('NativeUInt', 'Cardinal
7185:   Const('BitsPerByte','LongInt'( 8 );
7186:   Const('BitsPerWord','LongInt'( 16 );
7187:   Const('BitsPerLongWord','LongInt'( 32 );
7188: //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8 );
7189: //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8 );
7190: Function MinI( const A, B : Integer ) : Integer
7191: Function MaxI( const A, B : Integer ) : Integer
7192: Function MinC( const A, B : Cardinal ) : Cardinal
7193: Function MaxC( const A, B : Cardinal ) : Cardinal
7194: Function SumClipI( const A, I : Integer ) : Integer
7195: Function SumClipC( const A : Cardinal; const I : Integer ) : Cardinal
7196: Function InByteRange( const A : Int64 ) : Boolean
7197: Function InWordRange( const A : Int64 ) : Boolean
7198: Function InLongWordRange( const A : Int64 ) : Boolean
7199: Function InShortIntRange( const A : Int64 ) : Boolean
7200: Function InSmallIntRange( const A : Int64 ) : Boolean
7201: Function InLongIntRange( const A : Int64 ) : Boolean
7202: AddTypeS('Bool8', 'ByteBool'
7203: AddTypeS('Bool16', 'WordBool
7204: AddTypeS('Bool32', 'LongBool
7205: AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7206: AddTypeS('TCompareResultSet', 'set of TCompareResult
7207: Function ReverseCompareResult( const C : TCompareResult ) : TCompareResult
7208: Const('MinSingle','Single').setExtended( 1.5E-45 );
7209: Const('MaxSingle','Single').setExtended( 3.4E+38 );
7210: Const('MinDouble','Double').setExtended( 5.0E-324 );
7211: Const('MaxDouble','Double').setExtended( 1.7E+308 );
7212: Const('MinExtended','Extended').setExtended(3.4E-4932 );
7213: Const('MaxExtended','Extended').setExtended(1.1E+4932 );
7214: Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807 );
7215: Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807 );
7216: Function MinF( const A, B : Float ) : Float
7217: Function MaxF( const A, B : Float ) : Float
7218: Function ClipF( const Value : Float; const Low, High : Float ) : Float
7219: Function InSingleRange( const A : Float ) : Boolean
7220: Function InDoubleRange( const A : Float ) : Boolean
7221: Function InCurrencyRange( const A : Float ) : Boolean;
7222: Function InCurrencyRangel( const A : Int64 ) : Boolean;
7223: Function FloatExponentBase2( const A : Extended; var Exponent : Integer ) : Boolean
7224: Function FloatExponentBase10( const A : Extended; var Exponent : Integer ) : Boolean
7225: Function FloatIsInfinity( const A : Extended ) : Boolean
7226: Function FloatIsNaN( const A : Extended ) : Boolean
7227: Const('SingleCompareDelta','Extended').setExtended( 1.0E-34 );
7228: Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280 );
7229: Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400 );
7230: Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34 );
7231: Function FloatZero( const A : Float; const CompareDelta : Float ) : Boolean
7232: Function FloatOne( const A : Float; const CompareDelta : Float ) : Boolean
7233: Function FloatsEqual( const A, B : Float; const CompareDelta : Float ) : Boolean
7234: Function FloatsCompare( const A, B : Float; const CompareDelta : Float ) : TCompareResult
7235: Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5 );
7236: Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13 );
7237: Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17 );
7238: Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10 );
7239: Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double ) : Boolean
7240: Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult

```

```

7241: Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7242: Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7243: Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7244: Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7245: Function cIsHighBitSet( const Value : LongWord) : Boolean
7246: Function SetBitScanForward( const Value : LongWord) : Integer;
7247: Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7248: Function SetBitScanReverse( const Value : LongWord) : Integer;
7249: Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7250: Function ClearBitScanForward( const Value : LongWord) : Integer;
7251: Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7252: Function ClearBitScanReverse( const Value : LongWord) : Integer;
7253: Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7254: Function cReverseBits( const Value : LongWord) : LongWord;
7255: Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7256: Function cSwapEndian( const Value : LongWord) : LongWord
7257: Function cTwosComplement( const Value : LongWord) : LongWord
7258: Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7259: Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7260: Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7261: Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7262: Function cBitCount( const Value : LongWord) : LongWord
7263: Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7264: Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7265: Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7266: Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7267: Function SetbitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7268: Function ClearBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7269: Function ToggleBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7270: Function IsBitRangeSet( const Value: LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7271: Function IsBitRangeClear( const Value: LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7272: // AddTypeS('CharSet', 'set of AnsiChar'
7273: AddTypes('CharSet', 'set of Char' //!!!
7274: AddTypes('AnsiCharSet', 'TCharSet'
7275: AddTypes('ByteSet', 'set of Byte'
7276: AddTypes('AnsiChar', 'Char
7277: // Function AsCharSet( const C : array of AnsiChar) : CharSet
7278: Function AsByteSet( const C : array of Byte) : ByteSet
7279: Procedure ComplementChar( var C : CharSet; const Ch : Char)
7280: Procedure ClearCharSet( var C : CharSet)
7281: Procedure FillCharSet( var C : CharSet)
7282: procedure FillCharSearchRec; // with 0
7283: Procedure ComplementCharSet( var C : CharSet)
7284: Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7285: Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7286: Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7287: Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7288: Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7289: Function IsSubSet( const A, B : CharSet) : Boolean
7290: Function IsEqual( const A, B : CharSet) : Boolean
7291: Function IsEmpty( const C : CharSet) : Boolean
7292: Function IsComplete( const C : CharSet) : Boolean
7293: Function cCharCount( const C : CharSet) : Integer
7294: Procedure ConvertCaseInsensitive( var C : CharSet)
7295: Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7296: Function IntRangeLength( const Low, High : Integer) : Int64
7297: Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7298: Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7299: Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7300: Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7301: Function IntRangeIncludeElementRange(var Low, High:Integer;const LowElement,HighElement:Integer):Boolean
7302: Function CardRangeLength( const Low, High : Cardinal) : Int64
7303: Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7304: Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7305: Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7306: Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7307: Function CardRangeIncludeElementRange(var Low, High:Card;const LowElement,HighElement:Card):Boolean
7308: AddTypes('UnicodeChar', 'WideChar
7309: Function Compare( const I1, I2 : Boolean) : TCompareResult;
7310: Function CompareI( const I1, I2 : Integer) : TCompareResult;
7311: Function Compare2( const I1, I2 : Int64) : TCompareResult;
7312: Function Compare3( const I1, I2 : Extended) : TCompareResult;
7313: Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7314: Function CompareW( const I1, I2 : WideString) : TCompareResult
7315: Function cSgn( const A : LongInt) : Integer;
7316: Function cSgn1( const A : Int64) : Integer;
7317: Function cSgn2( const A : Extended) : Integer;
7318: AddTypes('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7319: Function AnsiCharToInt( const A : AnsiChar) : Integer
7320: Function WideCharToInt( const A : WideChar) : Integer
7321: Function CharToInt( const A : Char) : Integer
7322: Function IntToAnsiChar( const A : Integer) : AnsiChar
7323: Function IntToWideChar( const A : Integer) : WideChar
7324: Function IntToChar( const A : Integer) : Char
7325: Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7326: Function IsHexWideChar( const Ch : WideChar) : Boolean
7327: Function IsHexChar( const Ch : Char) : Boolean
7328: Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7329: Function HexWideCharToInt( const A : WideChar) : Integer

```

```

7330: Function HexCharToInt( const A : Char ) : Integer
7331: Function IntToUpperHexAnsiChar( const A : Integer ) : AnsiChar
7332: Function IntToUpperHexWideChar( const A : Integer ) : WideChar
7333: Function IntToUpperHexChar( const A : Integer ) : Char
7334: Function IntToLowerHexAnsiChar( const A : Integer ) : AnsiChar
7335: Function IntToLowerHexWideChar( const A : Integer ) : WideChar
7336: Function IntToLowerHexChar( const A : Integer ) : Char
7337: Function IntToStringA( const A : Int64 ) : AnsiString
7338: Function IntToStringW( const A : Int64 ) : WideString
7339: Function IntToString( const A : Int64 ) : String
7340: Function UIntToStringA( const A : NativeUInt ) : AnsiString
7341: Function UIntToStringW( const A : NativeUInt ) : WideString
7342: Function UIntToString( const A : NativeUInt ) : String
7343: Function LongWordToStrA( const A : LongWord; const Digits : Integer ) : AnsiString
7344: Function LongWordToStrW( const A : LongWord; const Digits : Integer ) : WideString
7345: Function LongWordToStrU( const A : LongWord; const Digits : Integer ) : UnicodeString
7346: Function LongWordToStr( const A : LongWord; const Digits : Integer ) : String
7347: Function LongWordToHexA( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7348: Function LongWordToHexW( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7349: Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7350: Function LongWordToOctA( const A : LongWord; const Digits : Integer ) : AnsiString
7351: Function LongWordToOctW( const A : LongWord; const Digits : Integer ) : WideString
7352: Function LongWordToOct( const A : LongWord; const Digits : Integer ) : String
7353: Function LongWordToBinA( const A : LongWord; const Digits : Integer ) : AnsiString
7354: Function LongWordToBinW( const A : LongWord; const Digits : Integer ) : WideString
7355: Function LongWordToBin( const A : LongWord; const Digits : Integer ) : String
7356: Function TryStringToInt64A( const S : AnsiString; out A : Int64 ) : Boolean
7357: Function TryStringToInt64W( const S : WideString; out A : Int64 ) : Boolean
7358: Function TryStringToInt64( const S : String; out A : Int64 ) : Boolean
7359: Function StringToInt64DefA( const S : AnsiString; const Default : Int64 ) : Int64
7360: Function StringToInt64DefW( const S : WideString; const Default : Int64 ) : Int64
7361: Function StringToInt64Def( const S : String; const Default : Int64 ) : Int64
7362: Function StringToInt64A( const S : AnsiString ) : Int64
7363: Function StringToInt64W( const S : WideString ) : Int64
7364: Function StringToInt64( const S : String ) : Int64
7365: Function TryStringToIntA( const S : AnsiString; out A : Integer ) : Boolean
7366: Function TryStringToIntW( const S : WideString; out A : Integer ) : Boolean
7367: Function TryStringToInt( const S : String; out A : Integer ) : Boolean
7368: Function StringToIntDefA( const S : AnsiString; const Default : Integer ) : Integer
7369: Function StringToIntDefW( const S : WideString; const Default : Integer ) : Integer
7370: Function StringToIntDef( const S : String; const Default : Integer ) : Integer
7371: Function StringToIntA( const S : AnsiString ) : Integer
7372: Function StringToIntW( const S : WideString ) : Integer
7373: Function StringToInt( const S : String ) : Integer
7374: Function TryStringToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7375: Function TryStringToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7376: Function TryStringToLongWord( const S : String; out A : LongWord ) : Boolean
7377: Function StringToLongWordA( const S : AnsiString ) : LongWord
7378: Function StringToLongWordW( const S : WideString ) : LongWord
7379: Function StringToLongWord( const S : String ) : LongWord
7380: Function HexToIntA( const S : AnsiString ) : NativeUInt
7381: Function HexToIntW( const S : WideString ) : NativeUInt
7382: Function HexToInt( const S : String ) : NativeUInt
7383: Function TryHexToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7384: Function TryHexToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7385: Function TryHexToLongWord( const S : String; out A : LongWord ) : Boolean
7386: Function HexToLongWordA( const S : AnsiString ) : LongWord
7387: Function HexToLongWordW( const S : WideString ) : LongWord
7388: Function HexToLongWord( const S : String ) : LongWord
7389: Function TryOctoToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7390: Function TryOctoToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7391: Function TryOctoToLongWord( const S : String; out A : LongWord ) : Boolean
7392: Function OctoToLongWordA( const S : AnsiString ) : LongWord
7393: Function OctoToLongWordW( const S : WideString ) : LongWord
7394: Function OctoToLongWord( const S : String ) : LongWord
7395: Function TryBinToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7396: Function TryBinToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7397: Function TryBinToLongWord( const S : String; out A : LongWord ) : Boolean
7398: Function BinToLongWordA( const S : AnsiString ) : LongWord
7399: Function BinToLongWordW( const S : WideString ) : LongWord
7400: Function BinToLongWord( const S : String ) : LongWord
7401: Function FloatToStringA( const A : Extended ) : AnsiString
7402: Function FloatToStringW( const A : Extended ) : WideString
7403: Function FloatToString( const A : Extended ) : String
7404: Function TryStringToFloatA( const A : AnsiString; out B : Extended ) : Boolean
7405: Function TryStringToFloatW( const A : WideString; out B : Extended ) : Boolean
7406: Function TryStringToFloat( const A : String; out B : Extended ) : Boolean
7407: Function StringToFloatA( const A : AnsiString ) : Extended
7408: Function StringToFloatW( const A : WideString ) : Extended
7409: Function StringToFloat( const A : String ) : Extended
7410: Function StringToFloatDefA( const A : AnsiString; const Default : Extended ) : Extended
7411: Function StringToFloatDefW( const A : WideString; const Default : Extended ) : Extended
7412: Function StringToFloatDef( const A : String; const Default : Extended ) : Extended
7413: Function EncodeBase64( const S, Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const PadChar: AnsiChar ) : AnsiString
7414: Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet ) : AnsiString
7415: unit uPSI_cFundamentUtils;
7416: Const ('b64_MIMEBase64','Str').String('ABCDEFIGHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/-._@ABCDEFGHIJKLM NOPQRSTUVWXYZ[\]^_');
7417: Const ('b64_UUEncode','String').String('!'#$%&'()'*+,.-./0123456789:;=>?@ABCDEFGHIJKLM NOPQRSTUVWXYZ[\]^_');

```

```

7418: Const ('b64_XXEncode', 'String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghi jklmnopqrstuvwxyz');
7419: Const ('CCHARSET', 'Stringb64_XXEncode');
7420: Const ('CHEXSET', 'String'0123456789ABCDEF
7421: Const ('HEXDIGITS', 'String'0123456789ABCDEF
7422: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7423: Const ('DIGISET', 'String'0123456789
7424: Const ('LETTERSET', 'String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7425: Const ('DIGISET2', 'TCharset').SetSet('0123456789'
7426: Const ('LETTERSET2', 'TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ')
7427: Const ('HEXSET2', 'TCharset').SetSet('0123456789ABCDEF');
7428: Const ('NUMBERSET', 'TCharset').SetSet('0123456789');
7429: Const ('NUMBERS', 'String'0123456789');
7430: Const ('LETTERS', 'String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7431: Function CharSetToStr( const C : CharSet) : AnsiString
7432: Function StrToCharSet( const S : AnsiString) : CharSet
7433: Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7434: Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7435: Function UUDecode( const S : AnsiString) : AnsiString
7436: Function XXDecode( const S : AnsiString) : AnsiString
7437: Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7438: Function InterfaceToStrA( const I : IInterface) : AnsiString
7439: Function InterfaceToStrW( const I : IInterface) : WideString
7440: Function InterfaceToStr( const I : IInterface) : String
7441: Function ObjectClassName( const O : TObject) : String
7442: Function ClassClassName( const C : TClass) : String
7443: Function ObjectToStr( const O : TObject) : String
7444: Function ObjectToString( const O : TObject) : String
7445: Function CharSetToStr( const C : CharSet) : AnsiString
7446: Function StrToCharSet( const S : AnsiString) : CharSet
7447: Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer; const AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7448: Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7449: Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7450: Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7451: Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7452: Const ('Bytes1KB','LongInt'( 1024));
7453: SIRegister_IInterface(CL);
7454: Procedure SelfTestCFundamentUtils
7455:
7456: Function CreateSchedule : IJclSchedule
7457: Function NullStamp : TTimeStamp
7458: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7459: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7460: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7461:
7462: procedure SIRegister_uwinplot(CL: TPSPPascalCompiler);
7463: begin
7464: AddTypeS('TFunc', 'function(X : Float) : Float;
7465: Function InitGraphics( Width, Height : Integer) : Boolean
7466: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7467: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7468: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7469: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7470: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7471: Procedure SetGraphTitle( Title : String)
7472: Procedure SetOxTitle( Title : String)
7473: Procedure SetOyTitle( Title : String)
7474: Function GetGraphTitle : String
7475: Function GetOxTitle : String
7476: Function GetOyTitle : String
7477: Procedure PlotOxAxis( Canvas : TCanvas)
7478: Procedure PlotOyAxis( Canvas : TCanvas)
7479: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7480: Procedure WriteGraphTitle( Canvas : TCanvas)
7481: Function SetMaxCurv( NCurv : Byte) : Boolean
7482: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7483: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7484: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7485: Procedure SetCurvStep( CurvIndex, Step : Integer)
7486: Function GetMaxCurv : Byte
7487: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7488: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7489: Function GetCurvLegend( CurvIndex : Integer) : String
7490: Function GetCurvStep( CurvIndex : Integer) : Integer
7491: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7492: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7493: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7494: Procedure PlotFunc(Canvas: TCanvas; Func: TFUnC; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7495: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7496: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7497: Function Xpixel( X : Float) : Integer
7498: Function Ypixel( Y : Float) : Integer
7499: Function Xuser( X : Integer) : Float
7500: Function Yuser( Y : Integer) : Float
7501: end;
7502:
7503: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)

```

```

7504: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7505: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7506: Procedure FFT_Integer_Cleanup
7507: Procedure CalcFrequency(NumSamples, FrequencyIndex: Integer; InArray: TCompVector; var FT : Complex)
7508: //unit uPSI_JclStreams;
7509: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7510: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7511: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7512: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7513:
7514: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7515: begin
7516:   FindClass('TOBJECT'), 'EInvalidDest'
7517:   FindClass('TOBJECT'), 'EFCantMove'
7518:   Procedure fmxCopyFile( const FileName, DestName : string)
7519:   Procedure fmxMoveFile( const FileName, DestName : string)
7520:   Function fmxGetFileSize( const FileName : string) : LongInt
7521:   Function fmxFileDateTime( const FileName : string) : TDateTime
7522:   Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7523:   Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7524: end;
7525:
7526: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7527: begin
7528:   SIRegister_IFindFileIterator(CL);
7529:   Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7530: end;
7531:
7532: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7533: begin
7534:   Function SkipWhite( cp : PChar) : PChar
7535:   Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7536:   Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7537:   Function ReadIdent( cp : PChar; var ident : string) : PChar
7538: end;
7539:
7540: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7541: begin
7542:   SIRegister_TStringHashMapTraits(CL);
7543:   Function CaseSensitiveTraits : TStringHashMapTraits
7544:   Function CaseInsensitiveTraits : TStringHashMapTraits
7545:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7546:   +'e; Right : PHashNode; end
7547: //PHashArray', '^THashArray // will not work
7548: SIRegister_TStringHashMap(CL);
7549: THashValue', 'Cardinal
7550: Function StrHash( const s : string) : THashValue
7551: Function TextHash( const s : string) : THashValue
7552: Function DataHash( var AValue, ASize : Cardinal) : THashValue
7553: Function Iterate_FreeObjects( AUUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7554: Function Iterate_Dispose( AUUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7555: Function Iterate_FreeMem( AUUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7556: SIRegister_TCaseSensitiveTraits(CL);
7557: SIRegister_TCaseInsensitiveTraits(CL);
7558:
7559: //*****unit uPSI_umath;
7560: Function uExpo( X : Float) : Float
7561: Function uExp2( X : Float) : Float
7562: Function uExp10( X : Float) : Float
7563: Function uLog( X : Float) : Float
7564: Function uLog2( X : Float) : Float
7565: Function uLog10( X : Float) : Float
7566: Function uLogA( X, A : Float) : Float
7567: Function uIntPower( X : Float; N : Integer): Float
7568: Function uPower( X, Y : Float) : Float
7569: Function SgnGamma( X : Float) : Integer
7570: Function Stirling( X : Float) : Float
7571: Function StirLog( X : Float) : Float
7572: Function Gamma( X : Float) : Float
7573: Function LnGamma( X : Float) : Float
7574: Function DiGamma( X : Float) : Float
7575: Function TriGamma( X : Float) : Float
7576: Function IGamma( X : Float) : Float
7577: Function JGamma( X : Float) : Float
7578: Function InvGamma( X : Float) : Float
7579: Function Erf( X : Float) : Float
7580: Function Erfc( X : Float) : Float
7581: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7582: { Correlation coefficient between samples X and Y }
7583: function DBeta(A, B, X : Float) : Float;
7584: { Density of Beta distribution with parameters A and B }
7585: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7586: Function Beta(X, Y : Float) : Float
7587: Function Binomial( N, K : Integer) : Float
7588: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7589: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7590: Procedure LU_Decompo( A : TMatrix; Lb, Ub : Integer)
7591: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)

```

```

7593: Function DNorm( X : Float ) : Float
7594:
7595: function DGamma(A, B, X : Float) : Float;
7596: { Density of Gamma distribution with parameters A and B }
7597: function DKhi2(Nu : Integer; X : Float) : Float;
7598: { Density of Khi-2 distribution with Nu d.o.f. }
7599: function DStudent(Nu : Integer; X : Float) : Float;
7600: { Density of Student distribution with Nu d.o.f. }
7601: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7602: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7603: function IBeta(A, B, X : Float) : Float;
7604: { Incomplete Beta function}
7605: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7606:
7607: procedure SIRegister_unlfit(CL: TPSPPascalCompiler);
7608: begin
7609:   Procedure SetOptAlgo( Algo : TOptAlgo)
7610:   procedure SetOptAlgo(Algo : TOptAlgo);
7611:   {
7612:     Sets the optimization algorithm according to Algo, which must be
7613:     NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ
7614:   }
7615:   Function GetOptAlgo : TOptAlgo
7616:   Procedure SetMaxParam( N : Byte)
7617:   Function GetMaxParam : Byte
7618:   Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7619:   Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7620:   Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7621:   Procedure NLFit( RegFunc: TRegFunc; DerivProc : TDervProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
7622:     Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7623:   Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDervProc; X, Y, S : TVector; Lb, Ub:Integer;
7624:     MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7625:   Procedure SetMCFile( FileName : String)
7626:   Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7627:   Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
7628:     LastPar:Integer;V:TMatrix);
7629: end;
7630: (*-----*)
7631: procedure SIRegister_usimplex(CL: TPSPPascalCompiler);
7632: begin
7633:   Procedure SaveSimplex( FileName : string)
7634:   Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7635: (*-----*)
7636: begin
7637:   Procedure RegTest(Y, Ycalc: TVector; LbY, UbY: Integer; V: TMatrix; LbV, UbV: Integer; var Test: TRegTest)
7638:   Procedure WRegTest(Y, Ycalc, S: TVector; LbY, UbY: Integer; V: TMatrix; LbV, UbV: Integer; var Test: TRegTest);
7639: end;
7640:
7641: procedure SIRegister_ustrings(CL: TPSPPascalCompiler);
7642: begin
7643:   Function LTrim( S : String) : String
7644:   Function RTrim( S : String) : String
7645:   Function uTrim( S : String) : String
7646:   Function StrChar( N : Byte; C : Char) : String
7647:   Function RFill( S : String; L : Byte) : String
7648:   Function LFill( S : String; L : Byte) : String
7649:   Function CFill( S : String; L : Byte) : String
7650:   Function Replace( S : String; C1, C2 : Char) : String
7651:   Function Extract( S : String; var Index : Byte; Delim : Char) : String
7652:   Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7653:   Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7654:   Function FloatStr( X : Float) : String
7655:   Function IntStr( N : LongInt) : String
7656:   Function uCompStr( Z : Complex) : String
7657: end;
7658:
7659: procedure SIRegister_uhyper(CL: TPSPPascalCompiler);
7660: begin
7661:   Function uSinh( X : Float ) : Float
7662:   Function uCosh( X : Float ) : Float
7663:   Function uTanh( X : Float ) : Float
7664:   Function uArcSinh( X : Float ) : Float
7665:   Function uArcCosh( X : Float ) : Float
7666:   Function ArcTanh( X : Float ) : Float
7667:   Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7668: end;
7669:
7670: procedure SIRegister_urandom(CL: TPSPPascalCompiler);
7671: begin
7672: type RNG_Type =
7673:   (RNG_MWC,      { Multiply-With-Carry }
7674:    RNG_MT,       { Mersenne Twister }
7675:    RNG_UVAG);   { Universal Virtual Array Generator }
7676:   Procedure SetRNG( RNG : RNG_Type)
7677:   Procedure InitGen( Seed : RNG_IntType)
7678:   Procedure SRand( Seed : RNG_IntType)

```

```

7679: Function IRanGen : RNG_IntType
7680: Function IRanGen31 : RNG_IntType
7681: Function RanGen1 : Float
7682: Function RanGen2 : Float
7683: Function RanGen3 : Float
7684: Function RanGen53 : Float
7685: end;
7686:
7687: // Optimization by Simulated Annealing
7688: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7689: begin
7690: Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7691: Procedure SA_CreateLogFile( FileName : String)
7692: Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7693: end;
7694:
7695: procedure SIRegister_urantuvg(CL: TPSPascalCompiler);
7696: begin
7697: Procedure InitUVAGbyString( KeyPhrase : string)
7698: Procedure InitUVAG( Seed : RNG_IntType)
7699: Function IRanUVAG : RNG_IntType
7700: end;
7701:
7702: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7703: begin
7704: Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7705: Procedure GA_CreateLogFile( LogFileName : String)
7706: Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7707: end;
7708:
7709: TVector', 'array of Float
7710: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7711: begin
7712: Procedure QSort( X : TVector; Lb, Ub : Integer)
7713: Procedure DQSort( X : TVector; Lb, Ub : Integer)
7714: end;
7715:
7716: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7717: begin
7718: Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7719: Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7720: end;
7721:
7722: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7723: begin
7724: FT_Result', 'Integer
7725: //TDWordptr', '^DWord // will not work
7726: TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7727: d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PChar'
7728: r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7729: ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeUp : Word; Rev4 : B'
7730: yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7731: te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7732: ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7733: erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7734: Current : Byte; BISHighCurrent : Byte; IFAISFifo : Byte; IFAISFifoTar : By'
7735: te; IFAISFastSer : Byte; AIsVCP : Byte; IFBISFifo : Byte; IFBISFifoTar : B'
7736: yte; IFBISFastSer : Byte; BISVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7737: Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7738: nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7739: ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 : '
7740: ; Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsvCP : B'
7741: yte; end
7742: end;
7743:
7744:
7745: //***** PaintFX*****
7746: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7747: begin
7748: //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7749: with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7750: Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7751: Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7752: Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7753: Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7754: Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7755: Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7756: Procedure Turn( Src, Dst : TBitmap)
7757: Procedure TurnRight( Src, Dst : TBitmap)
7758: Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7759: Procedure TexturizeFile( const Dst : TBitmap; Amount : Integer)
7760: Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7761: Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7762: Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7763: Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7764: Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7765: Procedure DrawMandelJulia(const Dst : TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7766: Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7767: Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)

```

```

7768: Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7769: Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7770: Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7771: Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7772: Procedure Emboss( var Bmp : TBitmap)
7773: Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7774: Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7775: Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7776: Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7777: Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7778: Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7779: Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7780: Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7781: Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7782: Procedure QuartoOpaque( Src, Dst : TBitmap)
7783: Procedure SemiOpaque( Src, Dst : TBitmap)
7784: Procedure ShadowDownLeft( const Dst : TBitmap)
7785: Procedure ShadowDownRight( const Dst : TBitmap)
7786: Procedure ShadowUpLeft( const Dst : TBitmap)
7787: Procedure ShadowUpRight( const Dst : TBitmap)
7788: Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7789: Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7790: Procedure FlipRight( const Dst : TBitmap)
7791: Procedure FlipDown( const Dst : TBitmap)
7792: Procedure Spotlight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7793: Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7794: Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7795: Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7796: Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7797: Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7798: Procedure SmoothResize( var Src, Dst : TBitmap)
7799: Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7800: Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7801: Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7802: Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7803: Procedure GrayScale( const Dst : TBitmap)
7804: Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7805: Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7806: Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7807: Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7808: Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7809: Procedure Spray( const Dst : TBitmap; Amount : Integer)
7810: Procedure AntiAlias( const Dst : TBitmap)
7811: Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7812: Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7813: Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7814: Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7815: Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7816: Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7817: Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7818: Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7819: Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7820: Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7821: Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7822: Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7823: Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7824: Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7825: Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7826: Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7827: Procedure Invert( Src : TBitmap)
7828: Procedure MirrorRight( Src : TBitmap)
7829: Procedure MirrorDown( Src : TBitmap)
7830: end;
7831: end;
7832:
7833: (*-----*)
7834: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7835: begin
7836:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7837:   +'ye, lbbrotate, lbtwist, lbrimble, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7838:   +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )'
7839:   SIRegister_TJvPaintFX(CL);
7840:   Function SplineFilter( Value : Single) : Single
7841:   Function BellFilter( Value : Single) : Single
7842:   Function TriangleFilter( Value : Single) : Single
7843:   Function BoxFilter( Value : Single) : Single
7844:   Function HermiteFilter( Value : Single) : Single
7845:   Function Lanczos3Filter( Value : Single) : Single
7846:   Function MitchellFilter( Value : Single) : Single
7847: end;
7848:
7849:
7850: (*-----*)
7851: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7852: begin
7853:   'TeeMsg_DefaultFunctionName', 'String 'TeeFunction
7854:   TeeMsg_DefaultSeriesName', 'String 'Series
7855:   TeeMsg_DefaultToolName', 'String 'ChartTool
7856:   ChartComponentPalette', 'String 'TeeChart

```

```

7857: TeeMaxLegendColumns', 'LongInt'( 2 );
7858: TeeDefaultLegendSymbolWidth', 'LongInt'( 20 );
7859: TeeTitleFootDistance, LongInt( 5 );
7860: SIRegister_TCustomChartWall(CL);
7861: SIRegister_TChartWall(CL);
7862: SIRegister_TChartLegendGradient(CL);
7863: TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7864: TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7865: FindClass('TOBJECT'), 'LegendException
7866: TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7867: + 'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7868: FindClass('TOBJECT'), 'TCustomChartLegend
7869: TLegendSymbolSize', '( lcsPercent, lcsPixels )
7870: TLegendSymbolPosition', '( spLeft, spRight )
7871: TSymbolDrawEvent', 'Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7872: TSymbolCalcHeight', 'Function : Integer
7873: SIRegister_TLegendSymbol(CL);
7874: SIRegister_TTeeCustomShapePosition(CL);
7875: TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7876: SIRegister_TLegendTitle(CL);
7877: SIRegister_TLegendItem(CL);
7878: SIRegister_TLegendItems(CL);
7879: TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7880: FindClass('TOBJECT'), 'TCustomChart
7881: SIRegister_TCustomChartLegend(CL);
7882: SIRegister_TChartLegend(CL);
7883: SIRegister_TChartTitle(CL);
7884: SIRegister_TChartFootTitle(CL);
7885: TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7886: + 'eButton; Shift : TShiftState; X, Y : Integer)
7887: TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7888: + 'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7889: TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series : '
7890: + TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7891: TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATile : '
7892: + 'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7893: TOnGetLegendPos', 'Procedure ( Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7894: TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7895: TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7896: + 'toMax : Boolean; Min : Double; Max : Double; end
7897: TAllAxisSavedScales', 'array of TAxisSavedScales
7898: SIRegister_TChartBackWall(CL);
7899: SIRegister_TChartRightWall(CL);
7900: SIRegister_TChartBottomWall(CL);
7901: SIRegister_TChartLeftWall(CL);
7902: SIRegister_TChartWalls(CL);
7903: TChartAllowScrollEvent', 'Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);'
7904: SIRegister_TCustomChart(CL);
7905: SIRegister_TChart(CL);
7906: SIRegister_TTeeSeriesTypes(CL);
7907: SIRegister_TTeeToolTypes(CL);
7908: SIRegister_TTeeDragObject(CL);
7909: SIRegister_TColorPalettes(CL);
7910: Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
    AGalleryPage:PString;ANumGallerySeries:Integer;
7911: Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADscription : PString);
7912: Procedure RegisterTeeFunction(AFuncClass:TTeeFunctionClass;ADescription,
    AGalleryPage:PString;ANumGallerySeries: Int;
7913: Procedure RegisterTeeBasicFunction(AFunctionClass : TTeeFunctionClass; ADscription : PString)
7914: Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
    ADscription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7915: Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7916: Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7917: Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7918: Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7919: Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
    AFunctionClass : TTeeFunctionClass) : TChartSeries
7920: Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7921: Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7922: Function CloneChartSeries2(TChartSeries:AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7923: Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent ) : TTeeCustomTool
7924: Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7925: Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7926: Function GetNewSeriesName( AOwner : TComponent ) : TComponentName
7927: Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7928: Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7929: Function GetGallerySeriesName( ASeries : TChartSeries ) : String
7930: Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
    R:TRect;ReferenceChart:TCustomChart);
7931: SIRegister_TChartTheme(CL);
7932: //TChartThemeClass', 'class of TChartTheme
7933: //TCanvasClass', 'class of TCanvas3D
7934: Function SeriesNameOrIndex( ASeries : TCustomChartSeries ) : String
7935: Function SeriesTitleOrName( ASeries : TCustomChartSeries ) : String
7936: Procedure FillSeriesItems( Items : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7937: Procedure ShowMessageUser( const S : String)
7938: Function HasNoMandatoryValues( ASeries : TChartSeries ) : Boolean
7939: Function HasLabels( ASeries : TChartSeries ) : Boolean
7940: Function HasColors( ASeries : TChartSeries ) : Boolean

```

```

7941: Function SeriesGuessContents( ASeries : TChartSeries ) : TeeFormatFlag
7942: Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7943: end;
7945:
7946: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7947: begin
7948: // 'TeeFormBorderStyle', 'bsNone';
7949: SIRegister_TMetafile(CL);
7950: 'TeeDefVerticalMargin','LongInt'( 4 );
7951: 'TeeDefHorizMargin','LongInt'( 3 );
7952: 'crTeeHand','LongInt'( TCursor( 2020 ) );
7953: 'TeeMsg_TeeHand','String 'crTeeHand
7954: 'TeeNormalPrintDetail','LongInt'( 0 );
7955: 'TeeHighPrintDetail','LongInt'( - 100 );
7956: 'TeeDefault_PrintMargin','LongInt'( 15 );
7957: 'MaxDefaultColors','LongInt'( 19 );
7958: 'TeeTabDelimiter','Char #9';
7959: TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7960: '+nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7961: +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7962: +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7963: +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7964: +'ourMonths, dtSixMonths, dtOneYear, dtNone )'
7965: SIRegister_TCustomPanelNoCaption(CL);
7966: FindClass('TOBJECT'), 'TCustomTeePanel
7967: SIRegister_TZoomPanning(CL);
7968: SIRegister_TTeeEvent(CL);
7969: //SIRegister_TTeeEventListeners(CL);
7970: TTeeMouseEventKind', '( meDown, meUp, meMove )'
7971: SIRegister_TTeeMouseEvent(CL);
7972: SIRegister_TCustomTeePanel(CL);
7973: //TChartGradient', 'TTeeGradient
7974: //TChartGradientClass', 'class of TChartGradient
7975: TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )'
7976: SIRegister_TTeeZoomPen(CL);
7977: SIRegister_TTeeZoomBrush(CL);
7978: TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )'
7979: SIRegister_TTeeZoom(CL);
7980: FindClass('TOBJECT'), 'TCustomTeePanelExtended
7981: TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )'
7982: SIRegister_TBackImage(CL);
7983: SIRegister_TCustomTeePanelExtended(CL);
7984: //TChartBrushClass', 'class of TChartBrush
7985: SIRegister_TTeeCustomShapeBrushPen(CL);
7986: TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )'
7987: TTextFormat , '( ttfNormal, ttfHtml )'
7988: SIRegister_TTeeCustomShape(CL);
7989: SIRegister_TTeeShape(CL);
7990: SIRegister_TTeeExportData(CL);
7991: Function TeeStr( const Num : Integer ) : String
7992: Function DateTimeDefaultFormat( const AStep : Double ) : String
7993: Function TEEDaysInMonth( Year, Month : Word ) : Word
7994: Function FindDateTimeStep( const StepValue : Double ) : TDateTimeStep
7995: Function NextDateTimeStep( const AStep : Double ) : Double
7996: Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer ) : Boolean;
7997: Function PointInLine1( const P, FromPoint, ToPoint : TPoint ) : Boolean;
7998: Function PointInLine2( const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7999: Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer ):Boolean;
8000: Function PointInLineTolerance( const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer ):Boolean;
8001: Function PointInPolygon( const P : TPoint; const Poly : array of TPoint ) : Boolean
8002: Function PointInTriangle( const P, P0, P1, P2 : TPoint ) : Boolean;
8003: Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer ) : Boolean;
8004: Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer ) : Boolean;
8005: Function PointInEllipse( const P : TPoint; const Rect : TRect ) : Boolean;
8006: Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer ) : Boolean;
8007: Function DelphiToLocalFormat( const Format : String ) : String
8008: Function LocalToDelphiFormat( const Format : String ) : String
8009: Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
8010: Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8011: Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
  AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
8012: TTeeSortCompare', 'Function ( a, b : Integer ) : Integer
8013: TTeeSortSwap', 'Procedure ( a, b : Integer )
8014: Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTEESortCompare;SwapFunc:TTEESortSwap);
8015: Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String ) : string
8016: Function TeeExtractField( St : String; Index : Integer ) : String;
8017: Function TeeExtractField1( St : String; Index : Integer; const Separator : String ) : String;
8018: Function TeeNumFields1( const St, Separator : String ) : Integer;
8019: Function TeeNumFields( St : String ) : Integer;
8020: Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8021: Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String )
8022: // TColorArray', 'array of TColor
8023: Function GetDefaultColor( const Index : Integer ) : TColor
8024: Procedure SetDefaultColorPalette;
8025: Procedure SetDefaultColorPalette1( const Palette : array of TColor );
8026: 'TeeCheckBoxSize','LongInt'( 11 );
8027: Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8028: Function TEEStrToFloatDef( const S : string; const Default : Extended ) : Extended

```

```

8029: Function TryStrToFloat( const S : String; var Value : Double) : Boolean
8030: Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
8031: Procedure TeeTranslateControl( AControl : TControl);
8032: Procedure TeeTranslateControl( AControl : TControl; const ExcludeChilds : array of TControl);
8033: Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
8034: //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8035: Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8036: //Procedure DrawBevel(Canvas:TeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8037: //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8038: Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8039: Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8040: Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8041: Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8042: Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8043: Procedure TeeSaveStringOption( const AKey, Value : String)
8044: Function TeeDefaultXMLEncoding : String
8045: Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8046: TeeWindowHandle', 'Integer
8047: Procedure TeeGoToURL( Handle : TeeWindowHandle; const URL : String)
8048: Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8049: Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8050: end;
8051:
8052:
8053: using mXBDEUtils
8054: ****
8055: Procedure SetAlias( aAlias, aDirectory : String)
8056: Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
Desired:Variant;Size:Byte);
8057: Function GetFileVersionNumber( const FileName : String) : TVersionNo
8058: Procedure SetBDE( aPath, aNode, aValue : String)
8059: function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8060: Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8061: Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8062: Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8063: Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8064: Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8065:
8066:
8067: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
8068: begin
8069: AddClassN(FindClass('TOBJECT'), 'EDateTime'
8070: Function DatePart( const D : TDateTime) : Integer
8071: Function TimePart( const D : TDateTime) : Double
8072: Function Century( const D : TDateTime) : Word
8073: Function Year( const D : TDateTime) : Word
8074: Function Month( const D : TDateTime) : Word
8075: Function Day( const D : TDateTime) : Word
8076: Function Hour( const D : TDateTime) : Word
8077: Function Minute( const D : TDateTime) : Word
8078: Function Second( const D : TDateTime) : Word
8079: Function Millisecond( const D : TDateTime) : Word
8080: ('OneDay','Extended').setExtended( 1.0);
8081: ('OneHour','Extended').SetExtended( OneDay / 24);
8082: ('OneMinute','Extended').SetExtended( OneHour / 60);
8083: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8084: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8085: ('OneWeek','Extended').SetExtended( OneDay * 7);
8086: ('HoursPerDay','Extended').SetExtended( 24);
8087: ('MinutesPerHour','Extended').SetExtended( 60);
8088: ('SecondsPerMinute','Extended').SetExtended( 60);
8089: Procedure SetYear( var D : TDateTime; const Year : Word)
8090: Procedure SetMonth( var D : TDateTime; const Month : Word)
8091: Procedure SetDay( var D : TDateTime; const Day : Word)
8092: Procedure SetHour( var D : TDateTime; const Hour : Word)
8093: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8094: Procedure SetSecond( var D : TDateTime; const Second : Word)
8095: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8096: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8097: Function IsEqual( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8098: Function IsEqual( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8099: Function IsAM( const D : TDateTime) : Boolean
8100: Function IsPM( const D : TDateTime) : Boolean
8101: Function IsMidnight( const D : TDateTime) : Boolean
8102: Function IsNoon( const D : TDateTime) : Boolean
8103: Function IsSunday( const D : TDateTime) : Boolean
8104: Function IsMonday( const D : TDateTime) : Boolean
8105: Function IsTuesday( const D : TDateTime) : Boolean
8106: Function IsWednesday( const D : TDateTime) : Boolean
8107: Function IsThursday( const D : TDateTime) : Boolean
8108: Function IsFriday( const D : TDateTime) : Boolean
8109: Function IsSaturday( const D : TDateTime) : Boolean
8110: Function IsWeekend( const D : TDateTime) : Boolean
8111: Function Noon( const D : TDateTime) : TDateTime
8112: Function Midnight( const D : TDateTime) : TDateTime
8113: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8114: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8115: Function NextWorkday( const D : TDateTime) : TDateTime
8116: Function PreviousWorkday( const D : TDateTime) : TDateTime

```

```

8117: Function FirstDayOfYear( const D : TDateTime ) : TDateTime
8118: Function LastDayOfYear( const D : TDateTime ) : TDateTime
8119: Function EasterSunday( const Year : Word ) : TDateTime
8120: Function GoodFriday( const Year : Word ) : TDateTime
8121: Function AddMilliseconds( const D : TDateTime; const N : Int64 ) : TDateTime
8122: Function AddSeconds( const D : TDateTime; const N : Int64 ) : TDateTime
8123: Function AddMinutes( const D : TDateTime; const N : Integer ) : TDateTime
8124: Function AddHours( const D : TDateTime; const N : Integer ) : TDateTime
8125: Function AddDays( const D : TDateTime; const N : Integer ) : TDateTime
8126: Function AddWeeks( const D : TDateTime; const N : Integer ) : TDateTime
8127: Function AddMonths( const D : TDateTime; const N : Integer ) : TDateTime
8128: Function AddYears( const D : TDateTime; const N : Integer ) : TDateTime
8129: Function DayOfYear( const Ye, Mo, Da : Word ) : Integer
8130: Function DayOfYear( const D : TDateTime ) : Integer
8131: Function DaysInMonth( const Ye, Mo : Word ) : Integer
8132: Function DaysInMonth( const D : TDateTime ) : Integer
8133: Function DaysInYear( const Ye : Word ) : Integer
8134: Function DaysInYearDate( const D : TDateTime ) : Integer
8135: Function WeekNumber( const D : TDateTime ) : Integer
8136: Function ISOFirstWeekOfYear( const Ye : Word ) : TDateTime
8137: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word )
8138: Function DiffMilliseconds( const D1, D2 : TDateTime ) : Int64
8139: Function DiffSeconds( const D1, D2 : TDateTime ) : Integer
8140: Function DiffMinutes( const D1, D2 : TDateTime ) : Integer
8141: Function DiffHours( const D1, D2 : TDateTime ) : Integer
8142: Function DiffDays( const D1, D2 : TDateTime ) : Integer
8143: Function DiffWeeks( const D1, D2 : TDateTime ) : Integer
8144: Function DiffMonths( const D1, D2 : TDateTime ) : Integer
8145: Function DiffYears( const D1, D2 : TDateTime ) : Integer
8146: Function GMTBias : Integer
8147: Function GMTTimeToLocalTime( const D : TDateTime ) : TDateTime
8148: Function LocalTimeToGMTTime( const D : TDateTime ) : TDateTime
8149: Function NowAsGMTTime : TDateTime
8150: Function DateTimeToISO8601String( const D : TDateTime ) : AnsiString
8151: Function ISO8601StringToTime( const D : AnsiString ) : TDateTime
8152: Function ISO8601StringAsDateTime( const D : AnsiString ) : TDateTime
8153: Function DateTimeToANSI( const D : TDateTime ) : Integer
8154: Function ANSIToDateTime( const Julian : Integer ) : TDateTime
8155: Function DateTimeToISOInteger( const D : TDateTime ) : Integer
8156: Function DateTimeToISOString( const D : TDateTime ) : AnsiString
8157: Function ISOIntegerToDate( const ISOInteger : Integer ) : TDateTime
8158: Function TwoDigitRadix2000YearToYear( const Y : Integer ) : Integer
8159: Function DateTimeAsElapsedTime( const D:TDateTime; const IncludeMilliseconds:Boolean ):AnsiString
8160: Function UnixTimeToDate( const UnixTime : LongWord ) : TDateTime
8161: Function DateTimeToUnixTime( const D : TDateTime ) : LongWord
8162: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer ) : AnsiString
8163: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer ) : UnicodeString
8164: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer ) : AnsiString
8165: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer ) : UnicodeString
8166: Function EnglishShortMonthStrA( const Month : Integer ) : AnsiString
8167: Function EnglishShortMonthStrU( const Month : Integer ) : UnicodeString
8168: Function EnglishLongMonthStrA( const Month : Integer ) : AnsiString
8169: Function EnglishLongMonthStrU( const Month : Integer ) : UnicodeString
8170: Function EnglishShortDayOfWeekA( const S : AnsiString ) : Integer
8171: Function EnglishShortDayOfWeekU( const S : UnicodeString ) : Integer
8172: Function EnglishLongDayOfWeekA( const S : AnsiString ) : Integer
8173: Function EnglishLongDayOfWeekU( const S : UnicodeString ) : Integer
8174: Function EnglishShortMonthA( const S : AnsiString ) : Integer
8175: Function EnglishShortMonthU( const S : UnicodeString ) : Integer
8176: Function EnglishLongMonthA( const S : AnsiString ) : Integer
8177: Function EnglishLongMonthU( const S : UnicodeString ) : Integer
8178: Function RFC850DayOfWeekA( const S : AnsiString ) : Integer
8179: Function RFC850DayOfWeekU( const S : UnicodeString ) : Integer
8180: Function RFC1123DayOfWeekA( const S : AnsiString ) : Integer
8181: Function RFC1123DayOfWeekU( const S : UnicodeString ) : Integer
8182: Function RFCMonthA( const S : AnsiString ) : Word
8183: Function RFCMonthU( const S : UnicodeString ) : Word
8184: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean ) : AnsiString
8185: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean ) : UnicodeString
8186: Function GMTDateTimeToRFC1123DateTimeA( const D : TDateTime; const IncludeDayOfWeek:Bool ):AnsiString;
8187: Function GMTDateTimeToRFC1123DateTimeU( const D:TDateTime;const IncludeDayOfWeek:Bool ):UnicodeString;
8188: Function DateTimeToRFCDate( const D : TDateTime ) : AnsiString
8189: Function DateTimeToRFCDate( const D : TDateTime ) : UnicodeString
8190: Function NowAsRFCDate : AnsiString
8191: Function NowAsRFCDate : UnicodeString
8192: Function RFCDate( const S : AnsiString ) : TDateTime
8193: Function RFCDate( const S : AnsiString ) : TDateTime
8194: Function RFCTimeZoneToGMTBias( const Zone : AnsiString ) : Integer
8195: Function TimePeriodStr( const D : TDateTime ) : AnsiString
8196: Procedure SelfTest
8197: end;
8198: //*****CFileUtils*****
8199: Function PathHasDriveLetterA( const Path : AnsiString ) : Boolean
8200: Function PathHasDriveLetter( const Path : String ) : Boolean
8201: Function PathIsDriveLetterA( const Path : AnsiString ) : Boolean
8202: Function PathIsDriveLetter( const Path : String ) : Boolean
8203: Function PathIsDriveRootA( const Path : AnsiString ) : Boolean
8204: Function PathIsDriveRoot( const Path : String ) : Boolean
8205: Function PathIsRootA( const Path : AnsiString ) : Boolean

```

```

8206: Function PathIsRoot( const Path : String ) : Boolean
8207: Function PathIsUNCPathA( const Path : AnsiString ) : Boolean
8208: Function PathIsUNCPath( const Path : String ) : Boolean
8209: Function PathIsAbsoluteA( const Path : AnsiString ) : Boolean
8210: Function PathIsAbsolute( const Path : String ) : Boolean
8211: Function PathIsDirectoryA( const Path : AnsiString ) : Boolean
8212: Function PathIsDirectory( const Path : String ) : Boolean
8213: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8214: Function PathInclSuffix( const Path : String; const PathSep : Char ) : String
8215: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8216: Function PathExclSuffix( const Path : String; const PathSep : Char ) : String
8217: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char )
8218: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char )
8219: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char )
8220: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char )
8221: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8222: Function PathCanonical( const Path : String; const PathSep : Char ) : String
8223: Function PathExpandA( const Path:AnsiString;const basePath:AnsiString;const PathSep:Char):AnsiString
8224: Function PathExpand( const Path : String; const basePath : String; const PathSep : Char ) : String
8225: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8226: Function PathLeftElement( const Path : String; const PathSep : Char ) : String
8227: Procedure PathSplitLeftElementA( const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char );
8228: Procedure PathSplitLeftElement( const Path:String; var LeftElement,RightPath: String;const PathSep:Char );
8229: Procedure DecodeFilePathA( const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char );
8230: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char )
8231: Function FileNameValidA( const FileName : AnsiString ) : AnsiString
8232: Function FileNameValid( const FileName : String ) : String
8233: Function FileNamePathA( const FileName,Path:AnsiString;const basePath:AnsiStr;const PathSep:Char):AnsiString;
8234: Function FilePath( const FileName, Path : String;const basePath: String;const PathSep : Char ) : String
8235: Function DirectoryExpandA( const Path:AnsiString;const basePath:AnsiString;const PathSep:Char ) : AnsiString
8236: Function DirectoryExpand( const Path : String; const basePath : String; const PathSep : Char ) : String
8237: Function UnixPathToWinPath( const Path : AnsiString ) : AnsiString
8238: Function WinPathToUnixPath( const Path : AnsiString ) : AnsiString
8239: Procedure CCopyFile( const FileName, DestName : String )
8240: Procedure CMoveFile( const FileName, DestName : String )
8241: Function CDeleteFiles( const FileMask : String ) : Boolean
8242: Function FileSeekEx( const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8243: Procedure FileCloseEx( const FileHandle : TFileHandle )
8244: Function FileExistsA( const FileName : AnsiString ) : Boolean
8245: Function CFileExists( const FileName : String ) : Boolean
8246: Function CFileGetSize( const FileName : String ) : Int64
8247: Function FileGetDateTime( const FileName : String ) : TDateTime
8248: Function FileGetDateTime2( const FileName : String ) : TDateTime
8249: Function FileIsReadOnly( const FileName : String ) : Boolean
8250: Procedure FileDeleteEx( const FileName : String )
8251: Procedure FileRenameEx( const OldFileName, NewFileName : String )
8252: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode : TFileCreationMode; const FileOpenWait : PFileOpenWait ) : AnsiString
8253: Function DirectoryEntryExists( const Name : String ) : Boolean
8254: Function DirectoryEntrySize( const Name : String ) : Int64
8255: Function CDirectoryExists( const DirectoryName : String ) : Boolean
8256: Function DirectoryGetDateTime( const DirectoryName : String ) : TDateTime
8257: Procedure CDirectoryCreate( const DirectoryName : String )
8258: Function GetFirstFileNameMatching( const FileMode : String ) : String
8259: Function DirEntryGetAttr( const FileName : AnsiString ) : Integer
8260: Function DirEntryIsDirectory( const FileName : AnsiString ) : Boolean
8261: Function FileHasAttr( const FileName : String; const Attr : Word ) : Boolean
8262: AddTypes('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote,
8263: +DriveCDRom, DriveRamDisk, DriveTypeUnknown )'
8264: Function DriveIsValid( const Drive : Char ) : Boolean
8265: Function DriveGetType( const Path : AnsiString ) : TLogicalDriveType
8266: Function DriveFreeSpace( const Path : AnsiString ) : Int64
8267:
8268: procedure SIRegister_cTimers(CL: TPPSPascalCompiler);
8269: begin
8270: AddClassN(FindClass('TOBJECT'),'ETimers
8271: Const ('TickFrequency', 'LongInt'( 1000):Function GetTick : LongWord
8272: Function TickDelta( const D1, D2 : LongWord ) : Integer
8273: Function TickDeltaW( const D1, D2 : LongWord ) : LongWord
8274: AddTypes('THPTimer', 'Int64
8275: Procedure StartTimer( var Timer : THPTimer )
8276: Procedure StopTimer( var Timer : THPTimer )
8277: Procedure ResumeTimer( var StoppedTimer : THPTimer )
8278: Procedure InitStoppedTimer( var Timer : THPTimer )
8279: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer )
8280: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean ) : Integer
8281: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean ) : Int64
8282: Procedure WaitMicroseconds( const MicroSeconds : Integer )
8283: Function GetHighPrecisionFrequency : Int64
8284: Function GetHighPrecisionTimerOverhead : Int64
8285: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64 )
8286: Procedure SelfTestCTimer
8287: end;
8288:
8289: procedure SIRegister_cRandom(CL: TPPSPascalCompiler);
8290: begin
8291: Function RandomSeed : LongWord
8292: Procedure AddEntropy( const Value : LongWord )
8293: Function RandomUniform : LongWord;

```

```

8294: Function RandomUniform1( const N : Integer ) : Integer;
8295: Function RandomBoolean : Boolean
8296: Function RandomByte : Byte
8297: Function RandomByteNonZero : Byte
8298: Function RandomWord : Word
8299: Function RandomInt64 : Int64;
8300: Function RandomInt641( const N : Int64 ) : Int64;
8301: Function RandomHex( const Digits : Integer ) : String
8302: Function RandomFloat : Extended
8303: Function RandomAlphaStr( const Length : Integer ) : AnsiString
8304: Function RandomPseudoWord( const Length : Integer ) : AnsiString
8305: Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8306: Function mwcRandomLongWord : LongWord
8307: Function urnRandomLongWord : LongWord
8308: Function moaRandomFloat : Extended
8309: Function mwcRandomFloat : Extended
8310: Function RandomNormalF : Extended
8311: Procedure SelfTestCRandom
8312: end;
8313:
8314: procedure SIRegister_SynEditMiscProcs(CL: TPSPPascalCompiler);
8315: begin
8316: // PIntArray', '^TIntArray // will not work
8317: Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8318: TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8319: Function synMax( x, y : integer ) : integer
8320: Function synMin( x, y : integer ) : integer
8321: Function synMinMax( x, mi, ma : integer ) : integer
8322: Procedure synSwapInt( var l, r : integer )
8323: Function synMaxPoint( const P1, P2 : TPoint ) : TPoint
8324: Function synMinPoint( const P1, P2 : TPoint ) : TPoint
8325: //Function synGetIntArray( Count : Cardinal; initialValue : integer ) : PIntArray
8326: Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect )
8327: Function synGetBestConvertTabsProc( TabWidth : integer ) : TConvertTabsProc
8328: Function synConvertTabs( const Line : AnsiString; TabWidth : integer ) : AnsiString
8329: Function synGetBestConvertTabsProcEx( TabWidth : integer ) : TConvertTabsProcEx
8330: Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8331: Function synGetExpandedLength( const aStr : string; aTabWidth : integer ) : integer
8332: Function synCharIndex2Caretpos( Index, TabWidth : integer; const Line : string ) : integer
8333: Function synCaretpos2CharIndex( TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8334: Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8335: Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8336: TStringType, '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat '
8337: + 'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )'
8338: ('C3_NONSPACING','LongInt'( 1 );
8339: 'C3_DIACRITIC','LongInt'( 2 );
8340: 'C3_VOWELMARK','LongInt'( 4 );
8341: ('C3_SYMBOL','LongInt'( 8 );
8342: ('C3_KATAKANA','LongWord( $0010 );
8343: ('C3_HIRAGANA','LongWord( $0020 );
8344: ('C3_HALFWIDTH','LongWord( $0040 );
8345: ('C3_FULLWIDTH','LongWord( $0080 );
8346: ('C3_IDEOGRAPH','LongWord( $0100 );
8347: ('C3_KASHIDA','LongWord( $0200 );
8348: ('C3_LEXICAL','LongWord( $0400 );
8349: ('C3_ALPHA','LongWord( $8000 );
8350: ('C3_NOTAPPLICABLE','LongInt'( 0 );
8351: Function synStrScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8352: Function synRScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8353: Function synIsStringType( Value : Word ) : TStringType
8354: Function synGetEOL( Line : PChar ) : PChar
8355: Function synEncodeString( s : string ) : string
8356: Function synDecodeString( s : string ) : string
8357: Procedure synFreeAndNil( var Obj: TObject )
8358: Procedure synAssert( Expr : Boolean )
8359: Function synLastDelimiter( const Delimiters, S : string ) : Integer
8360: TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8361: TReplaceFlags', 'set of TReplaceFlag )
8362: Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8363: Function synGetRValue( RGBValue : TColor ) : byte
8364: Function synGetGValue( RGBValue : TColor ) : byte
8365: Function synGetBValue( RGBValue : TColor ) : byte
8366: Function synRGB( r, g, b : Byte) : Cardinal
8367: // THighlighterAttriProc', 'Function( Highlighter : TSynCustomHigh'
8368: // +'lighter; Attr:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8369: //Function synEnumHighlighterAttrs( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
8370: HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer ) : Boolean
8371: Function synCalcFCS( const ABuf, ABufSize : Cardinal ) : Word
8372: Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
     AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8373: end;
8374: Function GET_APPCOMMAND_LPARAM( lParam : LPARAM ) : WORD
8375: Function GET_DEVICE_LPARAM( lParam : LPARAM ) : WORD
8376: Function GET_KEYSTATE_LPARAM( lParam : LPARAM ) : WORD
8377:
8378: procedure SIRegister_synthutil(CL: TPSPPascalCompiler);
8379: begin
8380: Function STimezoneBias : integer

```

```

8381: Function TimeZone : string
8382: Function Rfc822DateTime( t : TDateTime ) : string
8383: Function CDateTime( t : TDateTime ) : string
8384: Function SimpleDateTime( t : TDateTime ) : string
8385: Function AnsiCDatetime( t : TDateTime ) : string
8386: Function GetMonthNumber( Value : String ) : integer
8387: Function GetTimeFromStr( Value : string ) : TDateTime
8388: Function GetDateMDYFromStr( Value : string ) : TDateTime
8389: Function DecodeRfcDateTime( Value : string ) : TDateTime
8390: Function GetUTTIme : TDateTime
8391: Function SetUTTIme( Newdt : TDateTime ) : Boolean
8392: Function SGetTick : LongWord
8393: Function STickDelta( TickOld, TickNew : LongWord ) : LongWord
8394: Function CodeInt( Value : Word ) : Ansistring
8395: Function DecodeInt( const Value : Ansistring; Index : Integer ) : Word
8396: Function CodeLongInt( Value : LongInt ) : Ansistring
8397: Function DecodeLongInt( const Value : Ansistring; Index : Integer ) : LongInt
8398: Function DumpStr( const Buffer : Ansistring ) : string
8399: Function DumpExStr( const Buffer : Ansistring ) : string
8400: Procedure Dump( const Buffer : Ansistring; DumpFile : string )
8401: Procedure DumpEx( const Buffer : Ansistring; DumpFile : string )
8402: Function TrimSPLeft( const S : string ) : string
8403: Function TrimSPRight( const S : string ) : string
8404: Function TrimSP( const S : string ) : string
8405: Function SeparateLeft( const Value, Delimiter : string ) : string
8406: Function SeparateRight( const Value, Delimiter : string ) : string
8407: Function SGetParameter( const Value, Parameter : string ) : string
8408: Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings )
8409: Procedure ParseParameters( Value : string; const Parameters : TStrings )
8410: Function IndexByBegin( Value : string; const List : TStrings ) : integer
8411: Function GetEmailAddr( const Value : string ) : string
8412: Function GetEmailDesc( Value : string ) : string
8413: Function CStrToHex( const Value : Ansistring ) : string
8414: Function CIntToBin( Value : Integer; Digits : Byte ) : string
8415: Function CBinToInt( const Value : string ) : Integer
8416: Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string ):string
8417: Function CReplaceString( Value, Search, Replace : AnsiString ) : AnsiString
8418: Function CRPosEx( const Sub, Value : string; From : integer ) : Integer
8419: Function CRPos( const Sub, Value : String ) : Integer
8420: Function FetchBin( var Value : string; const Delimiter : string ) : string
8421: Function CFetch( var Value : string; const Delimiter : string ) : string
8422: Function FetchEx( var Value : string; const Delimiter, Quotation : string ) : string
8423: Function IsBinaryString( const Value : AnsiString ) : Boolean
8424: Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString ) : integer
8425: Procedure StringsTrim( const value : TStrings )
8426: Function PosFrom( const SubStr, Value : String; From : integer ) : integer
8427: Function IncPoint( const p : __pointer; Value : integer ) : __pointer
8428: Function GetBetween( const PairBegin, PairEnd, Value : string ) : string
8429: Function CCountOfChar( const Value : string; aChr : char ) : integer
8430: Function UnquoteStr( const Value : string; Quote : Char ) : string
8431: Function QuoteStr( const Value : string; Quote : Char ) : string
8432: Procedure HeadersToList( const Value : TStrings )
8433: Procedure ListToHeaders( const Value : TStrings )
8434: Function SwapBytes( Value : integer ) : integer
8435: Function ReadStrFromStream( const Stream : TStream; len : integer ) : AnsiString
8436: Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString )
8437: Function GetTempFile( const Dir, prefix : AnsiString ) : AnsiString
8438: Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar ) : AnsiString
8439: Function CXorString( Indata1, Indata2 : AnsiString ) : AnsiString
8440: Function NormalizeHeader( Value : TStrings; var Index : Integer ) : string
8441: end;
8442:
8443: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8444: begin
8445:   ('CrcBufSize', 'LongInt'( 2048 );
8446:   Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
8447:   Function Adler32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
8448:   Function Adler32OfFile( FileName : AnsiString ) : LongInt
8449:   Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
8450:   Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
8451:   Function Crc16OfFile( FileName : AnsiString ) : Cardinal
8452:   Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
8453:   Function Crc32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
8454:   Function Crc32OfFile( FileName : AnsiString ) : LongInt
8455:   Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
8456:   Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
8457:   Function InternetSumOfFile( FileName : AnsiString ) : Cardinal
8458:   Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
8459:   Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
8460:   Function Kermit16OfFile( FileName : AnsiString ) : Cardinal
8461: end;
8462:
8463: procedure SIRegister_ComObj(CL: TPSPascalCompiler);
8464: begin
8465:   function CreateOleObject(const ClassName: String): IDispatch;
8466:   function GetActiveOleObject(const ClassName: String): IDispatch;
8467:   function ProgIDToClassID(const ProgID: string): TGUID;
8468:   function ClassIDToProgID(const ClassID: TGUID): string;
8469:   function CreateClassID: string;

```

```

8470: function CreateGUIDString: string;
8471: function CreateGUIDID: string;
8472: procedure OleError(ErrorCode: longint);
8473: procedure OleCheck(Result: HResult);
8474: end;
8475:
8476: Function xCreateOleObject( const ClassName : string ) : Variant //or IDispatch
8477: Function xGetActiveOleObject( const ClassName : string ) : Variant
8478: //Function DlIGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj ) : HResult
8479: Function DllCanUnloadNow : HResult
8480: Function DllRegisterServer : HResult
8481: Function DllUnregisterServer : HResult
8482: Function VarFromInterface( Unknown : IUnknown ) : Variant
8483: Function VarToInterface( const V : Variant ) : IDispatch
8484: Function VarToAutoObject( const V : Variant ) : TAutoObject
8485: //Procedure
DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8486: //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo )
8487: Procedure OleError( ErrorCode : HResult )
8488: Procedure OleCheck( Result : HResult )
8489: Function StringToClassID( const S : string ) : TCLSID
8490: Function ClassIDToString( const ClassID : TCLSID ) : string
8491: Function xProgIDToClassID( const ProgID : string ) : TCLSID
8492: Function xClassIDToProgID( const ClassID : TCLSID ) : string
8493: Function xWideCompareStr( const S1, S2 : WideString ) : Integer
8494: Function xWideSameStr( const S1, S2 : WideString ) : Boolean
8495: Function xGUIDToString( const ClassID : TGUID ) : string
8496: Function xStringToGUID( const S : string ) : TGUID
8497: Function xGetModuleName( Module : HMODULE ) : string
8498: Function xAcquireExceptionObject : TObject
8499: Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer ) : Integer
8500: Function xUtf8Encode( const WS : WideString ) : UTF8String
8501: Function xUtf8Decode( const S : UTF8String ) : WideString
8502: Function xExcludeTrailingPathDelimiter( const S : string ) : string
8503: Function xIncludeTrailingPathDelimiter( const S : string ) : string
8504: Function XRTLHandleCOMException : HResult
8505: Procedure XRTLCheckArgument( Flag : Boolean )
8506: //Procedure XRTLCheckOutArgument( out Arg )
8507: Procedure XRTLInterfaceConnect( const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var Connection:Longint );
8508: Procedure XRTLInterfaceDisconnect( const Source: IUnknown; const IID:TIID;var Connection : Longint )
8509: Function XRTLRegisterActiveObject( const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var RegisterCookie:Int ) : HResult
8510: Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer ) : HResult
8511: //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj ) : HResult
8512: Procedure XRTLEnumActiveObjects( Strings : TStrings )
8513: function XRTLDefaultCategoryManager: IUnknown;
8514: function XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil) : Boolean;
8515: // ICatRegister helper functions
8516: function XRTLCREATEComponentCategory(CatID: TGUID; CatDescription: WideString;
8517:                                         LocaleID: TICLID = LOCALE_USER_DEFAULT;
8518:                                         const CategoryManager: IUnknown = nil) : HResult;
8519: function XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8520:                                         LocaleID: TICLID = LOCALE_USER_DEFAULT;
8521:                                         const CategoryManager: IUnknown = nil) : HResult;
8522: function XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8523:                                         const CategoryManager: IUnknown = nil) : HResult;
8524: function XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8525:                                         const CategoryManager: IUnknown = nil) : HResult;
8526: // ICatInformation helper functions
8527: function XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8528:                                         LocaleID: TICLID = LOCALE_USER_DEFAULT;
8529:                                         const CategoryManager: IUnknown = nil) : HResult;
8530: function XRTLGetCategoryList(Strings: TStrings; LocaleID: TICLID = LOCALE_USER_DEFAULT;
8531:                               const CategoryManager: IUnknown = nil) : HResult;
8532: function XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8533:                                     const CategoryManager: IUnknown = nil) : HResult;
8534: function XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8535:                                     const CategoryManager: IUnknown = nil) : HResult;
8536: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ');
8537:                                         const ADelete: Boolean = True): WideString;
8538: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8539: Function XRTLGetVariantAsString( const Value : Variant ) : string
8540: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone ) : TDateTime
8541: Function XRTLGetTimeZones : TXRTLTimeZones
8542: Function XFileTimeToDate( FileTime : TFileTime ) : TDateTime
8543: Function Date( Date : TDateTime ) : TFileTime
8544: Function GMTNow : TDateTime
8545: Function GMTToLocalTime( GMT : TDateTime ) : TDateTime
8546: Function LocalTimeToGMT( LocalTime : TDateTime ) : TDateTime
8547: Procedure XRTLNotImplemented
8548: Procedure XRTLRaiseError( E : Exception )
8549: Procedure XRTLRaise( E : Exception );
8550: Procedure XRaise( E : Exception );
8551: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string )
8552:
8553:
8554: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8555: begin

```

```

8556:     SIRegister_IXRTLValue(CL);
8557:     SIRegister_TXRTLValue(CL);
8558: //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8559: AddTypes('TXRTLValueArray', 'array of IXRTLValue
8560: Function XRTLValue( const AValue : Cardinal ) : IXRTLValue;
8561: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal ) : Cardinal;
8562: Function XRTLGetAsCardinal( const IValue : IXRTLValue ) : Cardinal;
8563: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal ) : Cardinal;
8564: Function XRTLValue1( const AValue : Integer ) : IXRTLValue;
8565: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer ) : Integer;
8566: Function XRTLGetAsInteger( const IValue : IXRTLValue ) : Integer;
8567: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer ) : Integer;
8568: Function XRTLValue2( const AValue : Int64 ) : IXRTLValue;
8569: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64 ) : Int64;
8570: Function XRTLGetAsInt64( const IValue : IXRTLValue ) : Int64;
8571: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64 ) : Int64;
8572: Function XRTLValue3( const AValue : Single ) : IXRTLValue;
8573: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single ) : Single;
8574: Function XRTLGetAssingle( const IValue : IXRTLValue ) : Single;
8575: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single ) : Single;
8576: Function XRTLValue4( const AValue : Double ) : IXRTLValue;
8577: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double ) : Double;
8578: Function XRTLGetAsDouble( const IValue : IXRTLValue ) : Double;
8579: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double ) : Double;
8580: Function XRTLValue5( const AValue : Extended ) : IXRTLValue;
8581: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended ) : Extended;
8582: Function XRTLGetAsExtended( const IValue : IXRTLValue ) : Extended;
8583: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended ) : Extended;
8584: Function XRTLValue6( const AValue : IInterface ) : IXRTLValue;
8585: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface ) : IInterface;
8586: Function XRTLGetAsInterface( const IValue : IXRTLValue ) : IInterface;
8587: //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj ) : IInterface;
8588: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface ) : IInterface;
8589: Function XRTLValue7( const AValue : WideString ) : IXRTLValue;
8590: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString ) : WideString;
8591: Function XRTLGetAsWideString( const IValue : IXRTLValue ) : WideString;
8592: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString ) : WideString;
8593: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean ) : IXRTLValue;
8594: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject ) : TObject;
8595: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean ) : TObject;
8596: Function XRTLGetAsObjectDef( const IValue : IXRTLValue; const DefValue : TObject; const
      ADetachOwnership : Boolean ) : TObject;
8597: //Function XRTLValue9( const AValue : _Pointer ) : IXRTLValue;
8598: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : _Pointer ) : _Pointer;
8599: //Function XRTLGetAsPointer( const IValue : IXRTLValue ) : _Pointer;
8600: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : _Pointer ) : _Pointer;
8601: Function XRTLValueV( const AValue : Variant ) : IXRTLValue;
8602: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant ) : Variant;
8603: Function XRTLGetAsVariant( const IValue : IXRTLValue ) : Variant;
8604: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant ) : Variant;
8605: Function XRTLValue10( const AValue : Currency ) : IXRTLValue;
8606: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency ) : Currency;
8607: Function XRTLGetAsCurrency( const IValue : IXRTLValue ) : Currency;
8608: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency ) : Currency;
8609: Function XRTLValue11( const AValue : Comp ) : IXRTLValue;
8610: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp ) : Comp;
8611: Function XRTLGetAsComp( const IValue : IXRTLValue ) : Comp;
8612: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp ) : Comp;
8613: Function XRTLValue12( const AValue : TClass ) : IXRTLValue;
8614: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass ) : TClass;
8615: Function XRTLGetAsClass( const IValue : IXRTLValue ) : TClass;
8616: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass ) : TClass;
8617: Function XRTLValue13( const AValue : TGUID ) : IXRTLValue;
8618: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID ) : TGUID;
8619: Function XRTLGetAsGUID( const IValue : IXRTLValue ) : TGUID;
8620: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID ) : TGUID;
8621: Function XRTLValue14( const AValue : Boolean ) : IXRTLValue;
8622: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean ) : Boolean;
8623: Function XRTLGetAsBoolean( const IValue : IXRTLValue ) : Boolean;
8624: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean ) : Boolean;
8625: end;
8626:
8627: //*****unit uPSI_GR32;*****
8628:
8629: Function Color32( WinColor : TColor ) : TColor32;
8630: Function Color321( R, G, B : Byte; A : Byte ) : TColor32;
8631: Function Color322( Index : Byte; var Palette : TPalette32 ) : TColor32;
8632: Function Gray32( Intensity : Byte; Alpha : Byte ) : TColor32;
8633: Function WinColor( Color32 : TColor32 ) : TColor;
8634: Function ArrayOfColor32( Colors : array of TColor32 ) : TArrayOfColor32;
8635: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte );
8636: Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte );
8637: Function Color32Components( R, G, B, A : Boolean ) : TColor32Components;
8638: Function RedComponent( Color32 : TColor32 ) : Integer;
8639: Function GreenComponent( Color32 : TColor32 ) : Integer;
8640: Function BlueComponent( Color32 : TColor32 ) : Integer;
8641: Function AlphaComponent( Color32 : TColor32 ) : Integer;
8642: Function Intensity( Color32 : TColor32 ) : Integer;
8643: Function SetAlpha( Color32 : TColor32; NewAlpha : Integer ) : TColor32;

```

```

8644: Function HSLtoRGB( H, S, L : Single ) : TColor32;
8645: Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single );
8646: Function HSLtoRGB1( H, S, L : Integer ) : TColor32;
8647: Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte );
8648: Function WinPalette( const P : TPalette32 ) : HPALETTE;
8649: Function FloatPoint( X, Y : Single ) : TFloatPoint;
8650: Function FloatPoint1( const P : TPoint ) : TFfloatPoint;
8651: Function FloatPoint2( const FXP : TFixedPoint ) : TFfloatPoint;
8652: Function FixedPoint( X, Y : Integer ) : TFfixedPoint;
8653: Function FixedPoint1( X, Y : Single ) : TFfixedPoint;
8654: Function FixedPoint2( const P : TPoint ) : TFfixedPoint;
8655: Function FixedPoint3( const FP : TFfloatPoint ) : TFfixedPoint;
8656: AddTypes('TRectRounding', '( rrClosest, rrOutside, rrInside )');
8657: Function MakeRect( const L, T, R, B : Integer ) : TRect;
8658: Function MakeRect1( const FR : TFfloatRect; Rounding : TRectRounding ) : TRect;
8659: Function MakeRect2( const FXR : TRect; Rounding : TRectRounding ) : TRect;
8660: Function GFixedRect( const L, T, R, B : TFixed ) : TRect;
8661: Function FixedRect1( const ARect : TRect ) : TRect;
8662: Function FixedRect2( const FR : TFfloatRect ) : TRect;
8663: Function GFloatRect( const L, T, R, B : TFloat ) : TFfloatRect;
8664: Function FloatRect1( const ARect : TRect ) : TFfloatRect;
8665: Function FloatRect2( const FXR : TRect ) : TFfloatRect;
8666: Function GIIntersectRect( out Dst : TRect; const R1, R2 : TRect ) : Boolean;
8667: Function IntersectRect( out Dst : TFfloatRect; const FR1, FR2 : TFfloatRect ) : Boolean;
8668: Function GUUnionRect( out Rect : TRect; const R1, R2 : TRect ) : Boolean;
8669: Function UnionRect1( out Rect : TFfloatRect; const R1, R2 : TFfloatRect ) : Boolean;
8670: Function GEEqualRect( const R1, R2 : TRect ) : Boolean;
8671: Function EqualRect1( const R1, R2 : TFfloatRect ) : Boolean;
8672: Procedure GIInflateRect( var R : TRect; Dx, Dy : Integer );
8673: Procedure InflateRect1( var FR : TFfloatRect; Dx, Dy : TFloat );
8674: Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer );
8675: Procedure OffsetRect1( var FR : TFfloatRect; Dx, Dy : TFloat );
8676: Function IsRectEmpty( const R : TRect ) : Boolean;
8677: Function IsRectEmpty1( const FR : TFfloatRect ) : Boolean;
8678: Function GPtInRect( const R : TRect; const P : TPoint ) : Boolean;
8679: Function PtInRect1( const R : TFfloatRect; const P : TPoint ) : Boolean;
8680: Function PtInRect2( const R : TRect; const P : TFfloatPoint ) : Boolean;
8681: Function PtInRect3( const R : TFfloatRect; const P : TFfloatPoint ) : Boolean;
8682: Function EqualRectSize( const R1, R2 : TRect ) : Boolean;
8683: Function EqualRectSize1( const R1, R2 : TFfloatRect ) : Boolean;
8684: Function MessageBeep( uType : UINT ) : BOOL;
8685: Function ShowCursor( bShow : BOOL ) : Integer;
8686: Function SetCursorPos( X, Y : Integer ) : BOOL;
8687: Function SetCursor( hCursor : HICON ) : HCURSOR;
8688: Function GetCursorPos( var lpPoint : TPoint ) : BOOL;
8689: //Function ClipCursor( lpRect : PRect ) : BOOL;
8690: Function GetClipCursor( var lpRect : TRect ) : BOOL;
8691: Function GetCursor : HCURSOR;
8692: Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer ) : BOOL;
8693: Function GetCaretBlinkTime : UINT;
8694: Function SetCaretBlinkTime( uMSeconds : UINT ) : BOOL;
8695: Function DestroyCaret : BOOL;
8696: Function HideCaret( hWnd : HWND ) : BOOL;
8697: Function ShowCaret( hWnd : HWND ) : BOOL;
8698: Function SetCaretPos( X, Y : Integer ) : BOOL;
8699: Function GetCaretPos( var lpPoint : TPoint ) : BOOL;
8700: Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint ) : BOOL;
8701: Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint ) : BOOL;
8702: Function MapWindowPoints( hWndFrom, hWndTo : HWND; var lpPoints, cPoints : UINT ) : Integer;
8703: Function WindowFromPoint( Point : TPoint ) : HWND;
8704: Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint ) : HWND;
8705:
8706:
8707: procedure SIRegister_GR32_Math(CL: TPPSPascalCompiler);
8708: begin
8709:   Function FixedFloor( A : TFixed ) : Integer;
8710:   Function FixedCeil( A : TFixed ) : Integer;
8711:   Function FixedMul( A, B : TFixed ) : TFixed;
8712:   Function FixedDiv( A, B : TFixed ) : TFixed;
8713:   Function OneOver( Value : TFixed ) : TFixed;
8714:   Function FixedRound( A : TFixed ) : Integer;
8715:   Function FixedSqr( Value : TFixed ) : TFixed;
8716:   Function FixedSqrtLP( Value : TFixed ) : TFixed;
8717:   Function FixedSqrtHP( Value : TFixed ) : TFixed;
8718:   Function FixedCombine( W, X, Y : TFixed ) : TFixed;
8719:   Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat );
8720:   Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single );
8721:   Function GRHypot( const X, Y : TFloat ) : TFloat;
8722:   Function Hypot1( const X, Y : Integer ) : Integer;
8723:   Function FastSqrt( const Value : TFloat ) : TFloat;
8724:   Function FastSqrtBab1( const Value : TFloat ) : TFloat;
8725:   Function FastSqrtBab2( const Value : TFloat ) : TFloat;
8726:   Function FastInvSqrt( const Value : Single ) : Single;
8727:   Function MulDiv( Multiplicand, Multiplier, Divisor : Integer ) : Integer;
8728:   Function GRIsPowerOf2( Value : Integer ) : Boolean;
8729:   Function PrevPowerOf2( Value : Integer ) : Integer;
8730:   Function NextPowerOf2( Value : Integer ) : Integer;
8731:   Function Average( A, B : Integer ) : Integer;
8732:   Function GRSign( Value : Integer ) : Integer;

```

```

8733: Function FloatMod( x, y : Double ) : Double
8734: end;
8735:
8736: procedure SIRegister_GR32_LowLevel(CL: TPSCompiler);
8737: begin
8738:   Function Clamp( const Value : Integer ) : Integer;
8739:   Procedure GRFillWord( var X, Count : Cardinal; Value : Longword )
8740:   Function StackAlloc( Size : Integer ) : Pointer
8741:   Procedure StackFree( P : Pointer )
8742:   Procedure Swap( var A, B : Pointer );
8743:   Procedure Swap1( var A, B : Integer );
8744:   Procedure Swap2( var A, B : TFixed );
8745:   Procedure Swap3( var A, B : TColor32 );
8746:   Procedure TestSwap( var A, B : Integer );
8747:   Procedure TestSwap1( var A, B : TFixed );
8748:   Function TestClip( var A, B : Integer; const Size : Integer ) : Boolean;
8749:   Function TestClip1( var A, B : Integer; const Start, Stop : Integer ) : Boolean;
8750:   Function GRConstrain( const Value, Lo, Hi : Integer ) : Integer;
8751:   Function Constrain1( const Value, Lo, Hi : Single ) : Single;
8752:   Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer ) : Integer
8753:   Function GRMin( const A, B, C : Integer ) : Integer;
8754:   Function GRMax( const A, B, C : Integer ) : Integer;
8755:   Function Clamp( Value, Max : Integer ) : Integer;
8756:   Function Clamp1( Value, Min, Max : Integer ) : Integer;
8757:   Function Wrap( Value, Max : Integer ) : Integer;
8758:   Function Wrap1( Value, Min, Max : Integer ) : Integer;
8759:   Function Wrap3( Value, Max : Single ) : Single;;
8760:   Function WrapPow2( Value, Max : Integer ) : Integer;
8761:   Function WrapPow21( Value, Min, Max : Integer ) : Integer;
8762:   Function Mirror( Value, Max : Integer ) : Integer;
8763:   Function Mirror1( Value, Min, Max : Integer ) : Integer;
8764:   Function MirrorPow2( Value, Max : Integer ) : Integer;
8765:   Function MirrorPow21( Value, Min, Max : Integer ) : Integer;
8766:   Function GetOptimalWrap( Max : Integer ) : TWrapProc;
8767:   Function GetOptimalWrap1( Min, Max : Integer ) : TWrapProcEx;
8768:   Function GetOptimalMirror( Max : Integer ) : TWrapProc;
8769:   Function GetOptimalMirror1( Min, Max : Integer ) : TWrapProcEx;
8770:   Function GetWrapProc( WrapMode : TWrapMode ) : TWrapProc;
8771:   Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer ) : TWrapProc;
8772:   Function GetWrapProcEx( WrapMode : TWrapMode ) : TWrapProcEx;
8773:   Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer ) : TWrapProcEx;
8774:   Function Div255( Value : Cardinal ) : Cardinal;
8775:   Function SAR_4( Value : Integer ) : Integer;
8776:   Function SAR_8( Value : Integer ) : Integer;
8777:   Function SAR_9( Value : Integer ) : Integer;
8778:   Function SAR_11( Value : Integer ) : Integer;
8779:   Function SAR_12( Value : Integer ) : Integer;
8780:   Function SAR_13( Value : Integer ) : Integer;
8781:   Function SAR_14( Value : Integer ) : Integer;
8782:   Function SAR_15( Value : Integer ) : Integer;
8783:   Function SAR_16( Value : Integer ) : Integer;
8784:   Function ColorSwap( WinColor : TColor ) : TColor32;
8785: end;
8786:
8787: procedure SIRegister_GR32_Filters(CL: TPSCompiler);
8788: begin
8789:   AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )');
8790:   Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components );
8791:   Procedure CopyComponents1(Dst:TCustomBmap32;DstX,DstY:Int32;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp);
8792:   Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32 );
8793:   Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean );
8794:   Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32 );
8795:   Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components );
8796:   Procedure InvertRGB( Dst, Src : TCustomBitmap32 );
8797:   Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean );
8798:   Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32 );
8799:   Function CreateBitmask( Components : TColor32Components ) : TColor32;
8800:   Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect; Bitmask : TColor32; LogicalOperator : TLogicalOperator );
8801:   Procedure ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8802:   Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean );
8803: end;
8804:
8805:
8806: procedure SIRegister_JclNTFS(CL: TPSCompiler);
8807: begin
8808:   AddClassN(FindClass('TOBJECT'), 'EJclNtfsError');
8809:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
8810:   Function NtfsGetCompression( const FileName : string; var State : Short ) : Boolean;
8811:   Function NtfsGetCompression1( const FileName : string ) : TFileCompressionState;
8812:   Function NtfsSetCompression( const FileName : string; const State : Short ) : Boolean;
8813:   Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState );
8814:   Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState );
8815:   Procedure NtfsSetDefaultFileCompression(const Directory: string; const State: TFileCompressionState );
8816:   Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState:Recursive:Boolean;
8817:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloc';
8818:   //'+tedRangeBuffer; MoreData : Boolean; end

```

```

8819: Function NtfsSetSparse( const FileName : string ) : Boolean
8820: Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64 ) : Boolean
8821: Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64 ) : Boolean
8822: //Function NtfsQueryAllocRanges(const FileName:string,Offset,Count:Int64,var
Ranges:TNtfsAllocRanges):Boolean;
8823: //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
Index:Integer):TFileAllocatedRangeBuffer
8824: Function NtfsSparseStreamsSupported( const Volume : string ) : Boolean
8825: Function NtfsGetSparse( const FileName : string ) : Boolean
8826: Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD ) : Boolean
8827: Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword ) : Boolean
8828: //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8829: Function NtfsGetReparseTag( const Path : string; var Tag : DWORD ) : Boolean
8830: Function NtfsReparsePointsSupported( const Volume : string ) : Boolean
8831: Function NtfsFileHasReparsePoint( const Path : string ) : Boolean
8832: Function NtfsIsFolderMountPoint( const Path : string ) : Boolean
8833: Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char ) : Boolean
8834: Function NtfsMountVolume( const Volume : Char; const MountPoint : string ) : Boolean
8835: AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )')
8836: Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8837: Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8838: Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8839: Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8840: Function NtfsRequestOpLock( Handle : THandle; Kind : TopLock; Overlapped : TOverlapped ) : Boolean
8841: Function NtfsCreateJunctionPoint( const Source, Destination : string ) : Boolean
8842: Function NtfsDeleteJunctionPoint( const Source : string ) : Boolean
8843: Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string ) : Boolean
8844: AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
+ 'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )')
8845: AddTypeS('TStreamIds', 'set of TStreamId')
8846: AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context :'
+ ': __Pointer; StreamIds : TStreamIds; end')
8847: AddTypes('TFindStreamData', 'record internal : TInternalFindStreamData; At'
+ 'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end')
8848: Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8849: Function NtfsFindNextStream( var Data : TFindStreamData ) : Boolean
8850: Function NtfsFindStreamClose( var Data : TFindStreamData ) : Boolean
8851: Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string ) : Boolean
8852: AddTypes('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end')
8853: Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo ) : Boolean
8854: Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
List:TStrings):Bool;
8855: Function NtfsDeleteHardLinks( const FileName : string ) : Boolean
8856: Function JclAppInstances : TJclAppInstances;
8857: Function JclAppInstances1( const UniqueAppIdGuidStr : string ) : TJclAppInstances;
8858: Procedure ReadMessageCheck( var Message : TMessage; const IgnoredOriginatorWnd : HWND ) : TJclAppInstDataKind
8859: Procedure ReadMessageData( const Message : TMessage; var Data : __Pointer; var Size : Integer )
8860: Procedure ReadMessageString( const Message : TMessage; var S : string )
8861: Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings )
8862: 
8863: 
8864: 
8865: 
8866: 
8867: (*-----*)
8868: procedure SIRegister_JclGraphics(CL: TPPascalCompiler);
8869: begin
8870:   FindClass('TOBJECT','EJclGraphicsError'
8871:             TDynDynIntegerArrayArray, 'array of TDynIntegerArray'
8872:             TDynPointArray, 'array of TPoint'
8873:             TDynDynPointArrayArray, 'array of TDynPointArray'
8874:             TPointF, 'record X : Single; Y : Single; end'
8875:             TDynPointArrayF, 'array of TPointF'
8876:             TDrawMode2, 'dmOpaque, dmBlend'
8877:             TStretchFilter2, '( sfNearest, sfLinear, sfSpline )'
8878:             TConversionKind, '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )'
8879:             TResamplingFilter, '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )'
8880:             TMatrix3d, 'record array[0..2,0..2] of extended end'
8881:             TDynDynPointArrayArrayF, 'array of TDynPointArrayF'
8882:             TScanLine, 'array of Integer'
8883:             TScanLines, 'array of TScanLine'
8884:             TColorChannel, '( ccRed, ccGreen, ccBlue, ccAlpha )'
8885:             TGradientDirection, '( gdVertical, gdHorizontal )'
8886:             TPolyFillMode, '( fmAlternate, fmWinding )'
8887:             TJclRegionCombineOperator, '( coAnd, coDiff, coOr, coXor )'
8888:             TJclRegionBitmapMode, '( rmInclude, rmExclude )'
8889:             TJclRegionKind, '( rkNull, rkSimple, rkComplex, rkError )'
8890:             SIRegister_TJclDesktopCanvas(CL);
8891:             FindClass('TOBJECT','TJclRegion'
8892:             SIRegister_TJclRegionInfo(CL);
8893:             SIRegister_TJclRegion(CL);
8894:             SIRegister_TJclThreadPersistent(CL);
8895:             SIRegister_TJclCustomMap(CL);
8896:             SIRegister_TJclBitmap32(CL);
8897:             SIRegister_TJclByteMap(CL);
8898:             SIRegister_TJclTransformation(CL);
8899:             SIRegister_TJclLinearTransformation(CL);
8900:             Procedure Stretch(NewWidth,
NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8901:             Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8902:             Procedure DrawBitmap( DC : HDC; Bitmap : HBitmap; X, Y, Width, Height : Integer)
8903:             Function GetAntialiasedBitmap( const Bitmap : TBitmap ) : TBitmap

```

```

8904: Procedure BitmapToJpeg( const FileName : string )
8905: Procedure JpegToBitmap( const FileName : string )
8906: Function ExtractIconCount( const FileName : string ) : Integer
8907: Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer ) : HICON
8908: Function IconToBitmapJ( Icon : HICON ) : HBITMAP
8909: Procedure
8910:   BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8911:   Procedure StretchTransfer(Dst:TJclBitmap32;
8912:     DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8913:   Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation )
8914:   Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect )
8915:   Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
8916:     TGradientDirection ) : Boolean;
8917:   Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
8918:     RegionBitmapMode:TJclRegionBitmapMode) : HGRN
8919:   Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND );
8920:   Procedure ScreenShot1( bm : TBitmap );
8921:   Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8922:   Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8923:   Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32 )
8924:   Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8925:   Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8926:   Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32 )
8927:   Procedure PolyPolygonsTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8928:   Procedure PolyPolygonsAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8929:   Procedure PolyPolygonsFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8930:   Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32 )
8931:   Procedure IntensityToAlpha( Dst, Src : TJclBitmap32 )
8932:   Procedure Invert( Dst, Src : TJclBitmap32 )
8933: end;
8934:
8935: (*-----*)
8936: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8937: begin
8938:   Function LockedAdd( var Target : Integer; Value : Integer ) : Integer
8939:   Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer ) : Integer
8940:   Function LockedCompareExchangeI( var Target : __Pointer; Exch, Comp : __Pointer ) : Pointer;
8941:   Function LockedDec( var Target : Integer ) : Integer
8942:   Function LockedExchange( var Target : Integer; Value : Integer ) : Integer
8943:   Function LockedExchangeAdd( var Target : Integer; Value : Integer ) : Integer
8944:   Function LockedExchangeDec( var Target : Integer ) : Integer
8945:   Function LockedExchangeInc( var Target : Integer ) : Integer
8946:   Function LockedExchangeSub( var Target : Integer; Value : Integer ) : Integer
8947:   Function LockedInc( var Target : Integer ) : Integer
8948:   Function LockedSub( var Target : Integer; Value : Integer ) : Integer
8949:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8950:   SIRegister_TJclDispatcherObject(CL);
8951:   Function WaitForMultipleObjects(const Objects:array of
8952:     TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8953:   Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
8954:     TimeOut : Cardinal):Cardinal
8955:   SIRegister_TJclCriticalSection(CL);
8956:   SIRegister_TJclEvent(CL);
8957:   SIRegister_TJclWaitableTimer(CL);
8958:   SIRegister_TJclSemaphore(CL);
8959:   SIRegister_TJclMutex(CL);
8960:   POptexSharedInfo', '^POptexSharedInfo // will not work
8961:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8962:   SIRegister_TJclOptex(CL);
8963:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8964:   TMrewThreadInfo', 'record Threadid : Longword; RecursionCount: Integer; Reader : Boolean; end
8965:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8966:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8967:   PMetSectSharedInfo', '^PMetSectSharedInfo // will not work
8968:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8969:     +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8970:   PMeteredSection', '^PMeteredSection // will not work
8971:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8972:   SIRegister_TJclMeteredSection(CL);
8973:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8974:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8975:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8976:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8977:   Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection ) : Boolean
8978:   Function QueryEvent( Handle : THandle; var Info : TEventInfo ) : Boolean
8979:   Function QueryMutex( Handle : THandle; var Info : TMutexInfo ) : Boolean
8980:   Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts ) : Boolean
8981:   Function QueryTimer( Handle : THandle; var Info : TTTimerInfo ) : Boolean
8982:   FindClass('TOBJECT'), EJclWin32HandleObjectError
8983:   FindClass('TOBJECT'), EJclDispatcherObjectError
8984:   FindClass('TOBJECT'), EJclCriticalSectionError
8985:   FindClass('TOBJECT'), EJclEventError
8986:   FindClass('TOBJECT'), EJclWaitableTimerError

```

```

8987:   FindClass( 'TOBJECT' ), 'EJclMutexError
8988:   FindClass( 'TOBJECT' ), 'EJclMeteredSectionError
8989: end;
8990:
8991:
8992: //*****unit uPSI_mORMotReport;
8993: Procedure SetCurrentPrinterAsDefault
8994: Function CurrentPrinterName : string
8995: Function mCurrentPrinterPaperSize : string
8996: Procedure UseDefaultPrinter
8997:
8998: procedure SIRegisterTSTREAM(C1: TPSPascalCompiler);
8999: begin
9000:   with FindClass( 'TOBJECT' ), 'TStream' ) do begin
9001:     IsAbstract := True;
9002:     //RegisterMethod('Function Read( var Buffer, Count : Longint ) : Longint
9003:     // RegisterMethod('Function Write( const Buffer, Count : Longint ) : Longint
9004:     function Read(Buffer:String;Count:LongInt):LongInt
9005:     function Write(Buffer:String;Count:LongInt):LongInt
9006:     function ReadString(Buffer:String;Count:LongInt):LongInt      //FileStream
9007:     function WriteString(Buffer:String;Count:LongInt):LongInt
9008:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
9009:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
9010:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt';
9011:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt';
9012:
9013:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
9014:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
9015:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
9016:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
9017:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
9018:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
9019:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
9020:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
9021:
9022:     function Seek(Offset:LongInt;Origin:Word):LongInt
9023:     procedure ReadBuffer(Buffer:String;Count:LongInt)
9024:     procedure WriteBuffer(Buffer:String;Count:LongInt)
9025:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)';
9026:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)';
9027:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)';
9028:     procedure WriteBufferFloat(Buffer:Double;Count:LongInt)';
9029:
9030:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
9031:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9032:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9033:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9034:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9035:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9036:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9037:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9038:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9039:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9040:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9041:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9042:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9043:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9044:
9045:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
9046:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
9047:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)';
9048:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)';
9049: //READBUFFERAC
9050:   function InstanceSize: Longint
9051:   Procedure FixupResourceHeader( FixupInfo : Integer )
9052:   Procedure ReadResHeader
9053:
9054: { $IFDEF DELPHI4UP }
9055:   function CopyFrom(Source:TStream;Count:Int64):LongInt
9056: { $ELSE }
9057:   function CopyFrom(Source:TStream;Count:Integer):LongInt
9058: { $ENDIF }
9059:   RegisterProperty('Position', 'LongInt', iptrw);
9060:   RegisterProperty('Size', 'LongInt', iptrw);
9061: end;
9062: end;
9063:
9064:
9065: { *****
9066: Unit DMATH - Interface for DMATH.DLL
9067: ***** }
9068: // see more docs/dmath_manual.pdf
9069:
9070: Function InitEval : Integer
9071: Procedure SetVariable( VarName : Char; Value : Float)
9072: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9073: Function Eval( ExpressionString : String) : Float
9074:
9075: unit dmath; //types are in built, others are external in DLL

```

```

9076: interface
9077: {$IFDEF DELPHI}
9078: uses
9079:   StdCtrls, Graphics;
9080: {$ENDIF}
9081: { -----
9082:   Types and constants
9083: ----- }
9084: {$i types.inc}
9085: { -----
9086:   Error handling
9087: ----- }
9088: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9089: { Sets the error code }
9090: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9091: { Sets error code and default function value }
9092: function MathErr : Integer; external 'dmath';
9093: { Returns the error code }
9094: { -----
9095:   Dynamic arrays
9096: ----- }
9097: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9098: { Sets the auto-initialization of arrays }
9099: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9100: { Creates floating point vector V[0..Ub] }
9101: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9102: { Creates integer vector V[0..Ub] }
9103: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9104: { Creates complex vector V[0..Ub] }
9105: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9106: { Creates boolean vector V[0..Ub] }
9107: procedure DimStrVector(var V : TStringVector; Ub : Integer); external 'dmath';
9108: { Creates string vector V[0..Ub] }
9109: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9110: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9111: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9112: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9113: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9114: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9115: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9116: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9117: procedure DimStrMatrix(var A : TStringMatrix; Ub1, Ub2 : Integer); external 'dmath';
9118: { Creates string matrix A[0..Ub1, 0..Ub2] }
9119: { -----
9120:   Minimum, maximum, sign and exchange
9121: ----- }
9122: function FMin(X, Y : Float) : Float; external 'dmath';
9123: { Minimum of 2 reals }
9124: function FMax(X, Y : Float) : Float; external 'dmath';
9125: { Maximum of 2 reals }
9126: function IMin(X, Y : Integer) : Integer; external 'dmath';
9127: { Minimum of 2 integers }
9128: function IMax(X, Y : Integer) : Integer; external 'dmath';
9129: { Maximum of 2 integers }
9130: function Sgn(X : Float) : Integer; external 'dmath';
9131: { Sign (returns 1 if X = 0) }
9132: function Sgn0(X : Float) : Integer; external 'dmath';
9133: { Sign (returns 0 if X = 0) }
9134: function DSgn(A, B : Float) : Float; external 'dmath';
9135: { Sgn(B) * |A| }
9136: procedure FSwap(var X, Y : Float); external 'dmath';
9137: { Exchange 2 reals }
9138: procedure ISwap(var X, Y : Integer); external 'dmath';
9139: { Exchange 2 integers }
9140: { -----
9141:   Rounding functions
9142: ----- }
9143: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9144: { Rounds X to N decimal places }
9145: function Ceil(X : Float) : Integer; external 'dmath';
9146: { Ceiling function }
9147: function Floor(X : Float) : Integer; external 'dmath';
9148: { Floor function }
9149: { -----
9150:   Logarithms, exponentials and power
9151: ----- }
9152: function Exp(X : Float) : Float; external 'dmath';
9153: { Exponential }
9154: function Exp2(X : Float) : Float; external 'dmath';
9155: { 2^X }
9156: function Exp10(X : Float) : Float; external 'dmath';
9157: { 10^X }
9158: function Log(X : Float) : Float; external 'dmath';
9159: { Natural log }
9160: function Log2(X : Float) : Float; external 'dmath';
9161: { Log, base 2 }
9162: function Log10(X : Float) : Float; external 'dmath';
9163: { Decimal log }
9164: function LogA(X, A : Float) : Float; external 'dmath';

```

```

9165: { Log, base A }
9166: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9167: { X^N }
9168: function Power(X, Y : Float) : Float; external 'dmath';
9169: { X^Y, X >= 0 }
9170: { -----
9171:   Trigonometric functions
9172: ----- }
9173: function Pythag(X, Y : Float) : Float; external 'dmath';
9174: { Sqrt(X^2 + Y^2) }
9175: function FixAngle(Theta : Float) : Float; external 'dmath';
9176: { Set Theta in -Pi..Pi }
9177: function Tan(X : Float) : Float; external 'dmath';
9178: { Tangent }
9179: function ArcSin(X : Float) : Float; external 'dmath';
9180: { Arc sinus }
9181: function ArcCos(X : Float) : Float; external 'dmath';
9182: { Arc cosinus }
9183: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9184: { Angle (Ox, OM) with M(X,Y) }
9185: { -----
9186:   Hyperbolic functions
9187: ----- }
9188: function Sinh(X : Float) : Float; external 'dmath';
9189: { Hyperbolic sine }
9190: function Cosh(X : Float) : Float; external 'dmath';
9191: { Hyperbolic cosine }
9192: function Tanh(X : Float) : Float; external 'dmath';
9193: { Hyperbolic tangent }
9194: function ArcSinh(X : Float) : Float; external 'dmath';
9195: { Inverse hyperbolic sine }
9196: function ArcCosh(X : Float) : Float; external 'dmath';
9197: { Inverse hyperbolic cosine }
9198: function ArcTanh(X : Float) : Float; external 'dmath';
9199: { Inverse hyperbolic tangent }
9200: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9201: { Sinh & Cosh }
9202: { -----
9203:   Gamma function and related functions
9204: ----- }
9205: function Fact(N : Integer) : Float; external 'dmath';
9206: { Factorial }
9207: function SgnGamma(X : Float) : Integer; external 'dmath';
9208: { Sign of Gamma function }
9209: function Gamma(X : Float) : Float; external 'dmath';
9210: { Gamma function }
9211: function LnGamma(X : Float) : Float; external 'dmath';
9212: { Logarithm of Gamma function }
9213: function Stirling(X : Float) : Float; external 'dmath';
9214: { Stirling's formula for the Gamma function }
9215: function StirLog(X : Float) : Float; external 'dmath';
9216: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9217: function DiGamma(X : Float) : Float; external 'dmath';
9218: { Digamma function }
9219: function TriGamma(X : Float) : Float; external 'dmath';
9220: { Trigamma function }
9221: function IGamma(A, X : Float) : Float; external 'dmath';
9222: { Incomplete Gamma function }
9223: function JGamma(A, X : Float) : Float; external 'dmath';
9224: { Complement of incomplete Gamma function }
9225: function InvGamma(A, P : Float) : Float; external 'dmath';
9226: { Inverse of incomplete Gamma function }
9227: function Erf(X : Float) : Float; external 'dmath';
9228: { Error function }
9229: function Erfc(X : Float) : Float; external 'dmath';
9230: { Complement of error function }
9231: { -----
9232:   Beta function and related functions
9233: ----- }
9234: function Beta(X, Y : Float) : Float; external 'dmath';
9235: { Beta function }
9236: function IBeta(A, B, X : Float) : Float; external 'dmath';
9237: { Incomplete Beta function }
9238: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9239: { Inverse of incomplete Beta function }
9240: { -----
9241:   Lambert's function
9242: ----- }
9243: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9244: { -----
9245:   Binomial distribution
9246: ----- }
9247: function Binomial(N, K : Integer) : Float; external 'dmath';
9248: { Binomial coefficient C(N,K) }
9249: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9250: { Probability of binomial distribution }
9251: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9252: { Cumulative probability for binomial distrib. }
9253: { -----

```

```

9254:   Poisson distribution
9255:   -----
9256: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9257: { Probability of Poisson distribution }
9258: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9259: { Cumulative probability for Poisson distrib. }
9260: { -----
9261:   Exponential distribution
9262:   -----
9263: function DExpo(A, X : Float) : Float; external 'dmath';
9264: { Density of exponential distribution with parameter A }
9265: function FExpo(A, X : Float) : Float; external 'dmath';
9266: { Cumulative probability function for exponential dist. with parameter A }
9267: { -----
9268:   Standard normal distribution
9269:   -----
9270: function DNorm(X : Float) : Float; external 'dmath';
9271: { Density of standard normal distribution }
9272: function FNorm(X : Float) : Float; external 'dmath';
9273: { Cumulative probability for standard normal distrib. }
9274: function PNorm(X : Float) : Float; external 'dmath';
9275: { Prob(|U| > X) for standard normal distrib. }
9276: function InvNorm(P : Float) : Float; external 'dmath';
9277: { Inverse of standard normal distribution }
9278: { -----
9279:   Student's distribution
9280:   -----
9281: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9282: { Density of Student distribution with Nu d.o.f. }
9283: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9284: { Cumulative probability for Student distrib. with Nu d.o.f. }
9285: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9286: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9287: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9288: { Inverse of Student's t-distribution function }
9289: { -----
9290:   Khi-2 distribution
9291:   -----
9292: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9293: { Density of Khi-2 distribution with Nu d.o.f. }
9294: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9295: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9296: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9297: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9298: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9299: { Inverse of Khi-2 distribution function }
9300: { -----
9301:   Fisher-Snedecor distribution
9302:   -----
9303: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9304: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9305: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9306: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9307: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9308: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9309: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9310: { Inverse of Snedecor's F-distribution function }
9311: { -----
9312:   Beta distribution
9313:   -----
9314: function DBeta(A, B, X : Float) : Float; external 'dmath';
9315: { Density of Beta distribution with parameters A and B }
9316: function FBeta(A, B, X : Float) : Float; external 'dmath';
9317: { Cumulative probability for Beta distrib. with param. A and B }
9318: { -----
9319:   Gamma distribution
9320:   -----
9321: function DGamma(A, B, X : Float) : Float; external 'dmath';
9322: { Density of Gamma distribution with parameters A and B }
9323: function FGamma(A, B, X : Float) : Float; external 'dmath';
9324: { Cumulative probability for Gamma distrib. with param. A and B }
9325: { -----
9326:   Expression evaluation
9327:   -----
9328: function InitEval : Integer; external 'dmath';
9329: { Initializes built-in functions and returns their number }
9330: function Eval(ExpressionString : String) : Float; external 'dmath';
9331: { Evaluates an expression at run-time }
9332: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9333: { Assigns a value to a variable }
9334: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9335: { Adds a function to the parser }
9336: { -----
9337:   Matrices and linear equations
9338:   -----
9339: procedure GaussJordan(A : TMatrix;
9340:                         Lb, Ubl, Ub2 : Integer;
9341:                         var Det : Float); external 'dmath';
9342: { Transforms a matrix according to the Gauss-Jordan method }

```

```

9343: procedure LinEq(A      : TMatrix;
9344:           B      : TVector;
9345:           Lb, Ub : Integer;
9346:           var Det : Float); external 'dmath';
9347: { Solves a linear system according to the Gauss-Jordan method }
9348: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9349: { Cholesky factorization of a positive definite symmetric matrix }
9350: procedure LU_Decom(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9351: { LU decomposition }
9352: procedure LU_Solve(A      : TMatrix;
9353:           B      : TVector;
9354:           Lb, Ub : Integer;
9355:           X      : TVector); external 'dmath';
9356: { Solution of linear system from LU decomposition }
9357: procedure QR_Decom(A      : TMatrix;
9358:           Lb, Ub1, Ub2 : Integer;
9359:           R      : TMatrix); external 'dmath';
9360: { QR decomposition }
9361: procedure QR_Solve(Q, R      : TMatrix;
9362:           B      : TVector;
9363:           Lb, Ub1, Ub2 : Integer;
9364:           X      : TVector); external 'dmath';
9365: { Solution of linear system from QR decomposition }
9366: procedure SV_Decom(A      : TMatrix;
9367:           Lb, Ub1, Ub2 : Integer;
9368:           S      : TVector;
9369:           V      : TMatrix); external 'dmath';
9370: { Singular value decomposition }
9371: procedure SV_SetZero(S      : TVector;
9372:           Lb, Ub : Integer;
9373:           Tol   : Float); external 'dmath';
9374: { Set lowest singular values to zero }
9375: procedure SV_Solve(U      : TMatrix;
9376:           S      : TVector;
9377:           V      : TMatrix;
9378:           B      : TVector;
9379:           Lb, Ub1, Ub2 : Integer;
9380:           X      : TVector); external 'dmath';
9381: { Solution of linear system from SVD }
9382: procedure SV_Approx(U      : TMatrix;
9383:           S      : TVector;
9384:           V      : TMatrix;
9385:           Lb, Ub1, Ub2 : Integer;
9386:           A      : TMatrix); external 'dmath';
9387: { Matrix approximation from SVD }
9388: procedure EigenVals(A      : TMatrix;
9389:           Lb, Ub : Integer;
9390:           Lambda : TCompVector); external 'dmath';
9391: { Eigenvalues of a general square matrix }
9392: procedure EigenVect(A      : TMatrix;
9393:           Lb, Ub : Integer;
9394:           Lambda : TCompVector;
9395:           V      : TMatrix); external 'dmath';
9396: { Eigenvalues and eigenvectors of a general square matrix }
9397: procedure EigenSym(A      : TMatrix;
9398:           Lb, Ub : Integer;
9399:           Lambda : TVector;
9400:           V      : TMatrix); external 'dmath';
9401: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9402: procedure Jacobi(A      : TMatrix;
9403:           Lb, Ub, MaxIter : Integer;
9404:           Tol   : Float;
9405:           Lambda : TVector;
9406:           V      : TMatrix); external 'dmath';
9407: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9408: { -----
9409: Optimization
9410: ----- }
9411: procedure MinBrack(Func      : TFunc;
9412:           var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9413: { Brackets a minimum of a function }
9414: procedure GoldSearch(Func      : TFunc;
9415:           A, B      : Float;
9416:           MaxIter   : Integer;
9417:           Tol       : Float;
9418:           var Xmin, Ymin : Float); external 'dmath';
9419: { Minimization of a function of one variable (golden search) }
9420: procedure LinMin(Func      : TFuncNVar;
9421:           X, DeltaX : TVector;
9422:           Lb, Ub : Integer;
9423:           var R : Float;
9424:           MaxIter : Integer;
9425:           Tol   : Float;
9426:           var F_min : Float); external 'dmath';
9427: { Minimization of a function of several variables along a line }
9428: procedure Newton(Func      : TFuncNVar;
9429:           HessGrad : THessGrad;
9430:           X      : TVector;
9431:           Lb, Ub : Integer;

```

```

9432:           MaxIter : Integer;
9433:           Tol : Float;
9434:           var F_min : Float;
9435:           G : TVector;
9436:           H_inv : TMatrix;
9437:           var Det : Float); external 'dmath';
9438: { Minimization of a function of several variables (Newton's method) }
9439: procedure SaveNewton(FileName : string); external 'dmath';
9440: { Save Newton iterations in a file }
9441: procedure Marquardt(Func : TFuncNVar;
9442:           HessGrad : THessGrad;
9443:           X : TVector;
9444:           Lb, Ub : Integer;
9445:           MaxIter : Integer;
9446:           Tol : Float;
9447:           var F_min : Float;
9448:           G : TVector;
9449:           H_inv : TMatrix;
9450:           var Det : Float); external 'dmath';
9451: { Minimization of a function of several variables (Marquardt's method) }
9452: procedure SaveMarquardt(FileName : string); external 'dmath';
9453: { Save Marquardt iterations in a file }
9454: procedure BFGS(Func : TFuncNVar;
9455:           Gradient : TGradient;
9456:           X : TVector;
9457:           Lb, Ub : Integer;
9458:           MaxIter : Integer;
9459:           Tol : Float;
9460:           var F_min : Float;
9461:           G : TVector;
9462:           H_inv : TMatrix); external 'dmath';
9463: { Minimization of a function of several variables (BFGS method) }
9464: procedure SaveBFGS(FileName : string); external 'dmath';
9465: { Save BFGS iterations in a file }
9466: procedure Simplex(Func : TFuncNVar;
9467:           X : TVector;
9468:           Lb, Ub : Integer;
9469:           MaxIter : Integer;
9470:           Tol : Float;
9471:           var F_min : Float); external 'dmath';
9472: { Minimization of a function of several variables (Simplex) }
9473: procedure SaveSimplex(FileName : string); external 'dmath';
9474: { Save Simplex iterations in a file }
9475: { -----
9476: Nonlinear equations
9477: ----- }
9478: procedure RootBrack(Func : TFunc;
9479:           var X, Y, FX, FY : Float); external 'dmath';
9480: { Brackets a root of function Func between X and Y }
9481: procedure Bisect(Func : TFunc;
9482:           var X, Y : Float;
9483:           MaxIter : Integer;
9484:           Tol : Float;
9485:           var F : Float); external 'dmath';
9486: { Bisection method }
9487: procedure Secant(Func : TFunc;
9488:           var X, Y : Float;
9489:           MaxIter : Integer;
9490:           Tol : Float;
9491:           var F : Float); external 'dmath';
9492: { Secant method }
9493: procedure NewtEq(Func, Deriv : TFunc;
9494:           var X : Float;
9495:           MaxIter : Integer;
9496:           Tol : Float;
9497:           var F : Float); external 'dmath';
9498: { Newton-Raphson method for a single nonlinear equation }
9499: procedure NewtEqs(Equations : TEquations;
9500:           Jacobian : TJacobian;
9501:           X, F : TVector;
9502:           Lb, Ub : Integer;
9503:           MaxIter : Integer;
9504:           Tol : Float); external 'dmath';
9505: { Newton-Raphson method for a system of nonlinear equations }
9506: procedure Broyden(Equations : TEquations;
9507:           X, F : TVector;
9508:           Lb, Ub : Integer;
9509:           MaxIter : Integer;
9510:           Tol : Float); external 'dmath';
9511: { Broyden's method for a system of nonlinear equations }
9512: { -----
9513: Polynomials and rational fractions
9514: ----- }
9515: function Poly(X : Float;
9516:           Coef : TVector;
9517:           Deg : Integer) : Float; external 'dmath';
9518: { Evaluates a polynomial }
9519: function Rfrac(X : Float;
9520:           Coef : TVector;

```

```

9521:           Deg1, Deg2 : Integer) : Float; external 'dmath';
9522: { Evaluates a rational fraction }
9523: function RootPol1(A, B : Float;
9524:                      var X : Float) : Integer; external 'dmath';
9525: { Solves the linear equation A + B * X = 0 }
9526: function RootPol2(Coef : TVector;
9527:                      Z : TCompVector) : Integer; external 'dmath';
9528: { Solves a quadratic equation }
9529: function RootPol3(Coef : TVector;
9530:                      Z : TCompVector) : Integer; external 'dmath';
9531: { Solves a cubic equation }
9532: function RootPol4(Coef : TVector;
9533:                      Z : TCompVector) : Integer; external 'dmath';
9534: { Solves a quartic equation }
9535: function RootPol(Coef : TVector;
9536:                      Deg : Integer;
9537:                      Z : TCompVector) : Integer; external 'dmath';
9538: { Solves a polynomial equation }
9539: function SetRealRoots(Deg : Integer;
9540:                         Z : TCompVector;
9541:                         Tol : Float) : Integer; external 'dmath';
9542: { Set the imaginary part of a root to zero }
9543: procedure SortRoots(Deg : Integer;
9544:                         Z : TCompVector); external 'dmath';
9545: { Sorts the roots of a polynomial }
9546: { -----
9547: Numerical integration and differential equations
9548: ----- }
9549: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9550: { Integration by trapezoidal rule }
9551: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9552: { Integral from A to B }
9553: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9554: { Integral from 0 to B }
9555: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9556: { Convolution product at time T }
9557: procedure ConvTrap(Func1, Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9558: { Convolution by trapezoidal rule }
9559: procedure RKF45(F : TDiffEqs;
9560:                     Neqn : Integer;
9561:                     Y, Yp : TVector;
9562:                     var T : Float;
9563:                     Tout, RelErr, AbsErr : Float;
9564:                     var Flag : Integer); external 'dmath';
9565: { Integration of a system of differential equations }
9566: { -----
9567: Fast Fourier Transform
9568: ----- }
9569: procedure FFT(NumSamples : Integer;
9570:                  InArray, OutArray : TCompVector); external 'dmath';
9571: { Fast Fourier Transform }
9572: procedure IFFT(NumSamples : Integer;
9573:                  InArray, OutArray : TCompVector); external 'dmath';
9574: { Inverse Fast Fourier Transform }
9575: procedure FFT_Integer(NumSamples : Integer;
9576:                         RealIn, ImagIn : TIntVector;
9577:                         OutArray : TCompVector); external 'dmath';
9578: { Fast Fourier Transform for integer data }
9579: procedure FFT_Integer_Cleanup; external 'dmath';
9580: { Clear memory after a call to FFT_Integer }
9581: procedure CalcFrequency(NumSamples,
9582:                         FrequencyIndex : Integer;
9583:                         InArray : TCompVector;
9584:                         var FFT : Complex); external 'dmath';
9585: { Direct computation of Fourier transform }
9586: { -----
9587: Random numbers
9588: ----- }
9589: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9590: { Select generator }
9591: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9592: { Initialize generator }
9593: function IRanGen : RNG_IntType; external 'dmath';
9594: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9595: function IRanGen31 : RNG_IntType; external 'dmath';
9596: { 31-bit random integer in [0 .. 2^31 - 1] }
9597: function RanGen1 : Float; external 'dmath';
9598: { 32-bit random real in [0,1] }
9599: function RanGen2 : Float; external 'dmath';
9600: { 32-bit random real in [0,1) }
9601: function RanGen3 : Float; external 'dmath';
9602: { 32-bit random real in (0,1) }
9603: function RanGen53 : Float; external 'dmath';
9604: { 53-bit random real in [0,1) }
9605: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9606: { Initializes the 'Multiply with carry' random number generator }
9607: function IRanMWC : RNG_IntType; external 'dmath';
9608: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9609: procedure InitMT(Seed : RNG_IntType); external 'dmath';

```

```

9610: { Initializes Mersenne Twister generator with a seed }
9611: procedure InitMTbyArray(InitKey : array of RNG_LongType;
9612:                           KeyLength : Word); external 'dmath';
9613: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9614: function IRanMT : RNG_IntType; external 'dmath';
9615: { Random integer from MT generator }
9616: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9617: { Initializes the UVAG generator with a string }
9618: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9619: { Initializes the UVAG generator with an integer }
9620: function IRanUVAG : RNG_IntType; external 'dmath';
9621: { Random integer from UVAG generator }
9622: function RanGaussStd : Float; external 'dmath';
9623: { Random number from standard normal distribution }
9624: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9625: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9626: procedure RanMult(M : TVector; L : TMatrix;
9627:                      Lb, Ub : Integer;
9628:                      X : TVector); external 'dmath';
9629: { Random vector from multinormal distribution (correlated) }
9630: procedure RanMultIndep(M, S : TVector;
9631:                          Lb, Ub : Integer;
9632:                          X : TVector); external 'dmath';
9633: { Random vector from multinormal distribution (uncorrelated) }
9634: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9635: { Initializes Metropolis-Hastings parameters }
9636: procedure GetMHParams(var NCycles, MaxSim, SavedSim:Integer); external 'dmath';
9637: { Returns Metropolis-Hastings parameters }
9638: procedure Hastings(Func : TFuncNVar;
9639:                        T : Float;
9640:                        X : TVector;
9641:                        V : TMatrix;
9642:                        Lb, Ub : Integer;
9643:                        Xmat : TMatrix;
9644:                        X_min : TVector;
9645:                        var F_min : Float); external 'dmath';
9646: { Simulation of a probability density function by Metropolis-Hastings }
9647: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9648: { Initializes Simulated Annealing parameters }
9649: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9650: { Initializes log file }
9651: procedure SimAnn(Func : TFuncNVar;
9652:                      X, Xmin, Xmax : TVector;
9653:                      Lb, Ub : Integer;
9654:                      var F_min : Float); external 'dmath';
9655: { Minimization of a function of several var. by simulated annealing }
9656: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9657: { Initializes Genetic Algorithm parameters }
9658: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9659: { Initializes log file }
9660: procedure GenAlg(Func : TFuncNVar;
9661:                      X, Xmin, Xmax : TVector;
9662:                      Lb, Ub : Integer;
9663:                      var F_min : Float); external 'dmath';
9664: { Minimization of a function of several var. by genetic algorithm }
9665: { -----
9666: Statistics
9667: -----
9668: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9669: { Mean of sample X }
9670: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9671: { Minimum of sample X }
9672: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9673: { Maximum of sample X }
9674: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9675: { Median of sample X }
9676: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9677: { Standard deviation estimated from sample X }
9678: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9679: { Standard deviation of population }
9680: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9681: { Correlation coefficient }
9682: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9683: { Skewness of sample X }
9684: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9685: { Kurtosis of sample X }
9686: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9687: { Quick sort (ascending order) }
9688: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9689: { Quick sort (descending order) }
9690: procedure Interval(X1, X2 : Float;
9691:                       MinDiv, MaxDiv : Integer;
9692:                       var Min, Max, Step : Float); external 'dmath';
9693: { Determines an interval for a set of values }
9694: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9695:                       var XMin, XMax, XStep : Float); external 'dmath';
9696: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9697: procedure StudIndep(N1, N2 : Integer;
9698:                       M1, M2, S1, S2 : Float);

```

```

9699:           var T          : Float;
9700:           var DoF         : Integer); external 'dmath';
9701: { Student t-test for independent samples }
9702: procedure StudPaired(X, Y      : TVector;
9703:                         Lb, Ub   : Integer;
9704:                         var T    : Float;
9705:                         var DoF  : Integer); external 'dmath';
9706: { Student t-test for paired samples }
9707: procedure AnOVal(Ns        : Integer;
9708:                      N         : TIntVector;
9709:                      M, S     : TVector;
9710:                      var V_f, V_r, F : Float;
9711:                      var DoF_f, DoF_r : Integer); external 'dmath';
9712: { One-way analysis of variance }
9713: procedure AnOva2(NA, NB, Nobs : Integer;
9714:                      M, S     : TMatrix;
9715:                      V, F     : TVector;
9716:                      DoF      : TIntVector); external 'dmath';
9717: { Two-way analysis of variance }
9718: procedure Snedecor(N1, N2      : Integer;
9719:                         S1, S2   : Float;
9720:                         var F    : Float;
9721:                         var DoF1, DoF2 : Integer); external 'dmath';
9722: { Snedecor's F-test (comparison of two variances) }
9723: procedure Bartlett(Ns       : Integer;
9724:                         N         : TIntVector;
9725:                         S         : TVector;
9726:                         var Khi2  : Float;
9727:                         var DoF   : Integer); external 'dmath';
9728: { Bartlett's test (comparison of several variances) }
9729: procedure Mann_Whitney(N1, N2      : Integer;
9730:                         X1, X2   : TVector;
9731:                         var U, Eps : Float); external 'dmath';
9732: { Mann-Whitney test }
9733: procedure Wilcoxon(X, Y      : TVector;
9734:                         Lb, Ub   : Integer;
9735:                         var Ndiff : Integer;
9736:                         var T, Eps : Float); external 'dmath';
9737: { Wilcoxon test }
9738: procedure Kruskal_Wallis(Ns     : Integer;
9739:                           N         : TIntVector;
9740:                           X         : TMatrix;
9741:                           var H    : Float;
9742:                           var DoF  : Integer); external 'dmath';
9743: { Kruskal-Wallis test }
9744: procedure Khi2_Conform(N_cls  : Integer;
9745:                           N_estim : Integer;
9746:                           Obs     : TIntVector;
9747:                           Calc    : TVector;
9748:                           var Khi2 : Float;
9749:                           var DoF  : Integer); external 'dmath';
9750: { Khi-2 test for conformity }
9751: procedure Khi2_Indep(N_lin   : Integer;
9752:                           N_col   : Integer;
9753:                           Obs     : TIntMatrix;
9754:                           var Khi2 : Float;
9755:                           var DoF  : Integer); external 'dmath';
9756: { Khi-2 test for independence }
9757: procedure Woolf_Conform(N_cls  : Integer;
9758:                           N_estim : Integer;
9759:                           Obs     : TIntVector;
9760:                           Calc    : TVector;
9761:                           var G    : Float;
9762:                           var DoF  : Integer); external 'dmath';
9763: { Woolf's test for conformity }
9764: procedure Woolf_Indep(N_lin   : Integer;
9765:                           N_col   : Integer;
9766:                           Obs     : TIntMatrix;
9767:                           var G    : Float;
9768:                           var DoF  : Integer); external 'dmath';
9769: { Woolf's test for independence }
9770: procedure DimStatClassVector(var C : TStatClassVector;
9771:                                 Ub    : Integer); external 'dmath';
9772: { Allocates an array of statistical classes: C[0..Ub] }
9773: procedure Distrib(X      : TVector;
9774:                       Lb, Ub   : Integer;
9775:                       A, B, H : Float;
9776:                       C       : TStatClassVector); external 'dmath';
9777: { Distributes an array X[Lb..Ub] into statistical classes }
9778: { -----
9779:  Linear / polynomial regression
9780:  ----- }
9781: procedure LinFit(X, Y      : TVector;
9782:                       Lb, Ub   : Integer;
9783:                       B       : TVector;
9784:                       V       : TMatrix); external 'dmath';
9785: { Linear regression : Y = B(0) + B(1) * X }
9786: procedure WLinFit(X, Y, S : TVector;
9787:                       Lb, Ub   : Integer;

```

```

9788:           B      : TVector;
9789:           V      : TMatrix); external 'dmath';
9790: { Weighted linear regression : Y = B(0) + B(1) * X }
9791: procedure SVDLinFit(X, Y : TVector;
9792:                       Lb, Ub : Integer;
9793:                       SVDTol : Float;
9794:                       B      : TVector;
9795:                       V      : TMatrix); external 'dmath';
9796: { Unweighted linear regression by singular value decomposition }
9797: procedure WSVDLinFit(X, Y, S : TVector;
9798:                        Lb, Ub : Integer;
9799:                        SVDTol : Float;
9800:                        B      : TVector;
9801:                        V      : TMatrix); external 'dmath';
9802: { Weighted linear regression by singular value decomposition }
9803: procedure MulFit(X      : TMatrix;
9804:                     Y      : TVector;
9805:                     Lb, Ub, Nvar : Integer;
9806:                     ConsTerm : Boolean;
9807:                     B      : TVector;
9808:                     V      : TMatrix); external 'dmath';
9809: { Multiple linear regression by Gauss-Jordan method }
9810: procedure WMulFit(X      : TMatrix;
9811:                      Y, S   : TVector;
9812:                      Lb, Ub, Nvar : Integer;
9813:                      ConsTerm : Boolean;
9814:                      B      : TVector;
9815:                      V      : TMatrix); external 'dmath';
9816: { Weighted multiple linear regression by Gauss-Jordan method }
9817: procedure SVDFit(X      : TMatrix;
9818:                     Y      : TVector;
9819:                     Lb, Ub, Nvar : Integer;
9820:                     ConsTerm : Boolean;
9821:                     SVDTol : Float;
9822:                     B      : TVector;
9823:                     V      : TMatrix); external 'dmath';
9824: { Multiple linear regression by singular value decomposition }
9825: procedure WSVDFit(X      : TMatrix;
9826:                      Y, S   : TVector;
9827:                      Lb, Ub, Nvar : Integer;
9828:                      ConsTerm : Boolean;
9829:                      SVDTol : Float;
9830:                      B      : TVector;
9831:                      V      : TMatrix); external 'dmath';
9832: { Weighted multiple linear regression by singular value decomposition }
9833: procedure PolFit(X, Y      : TVector;
9834:                      Lb, Ub, Deg : Integer;
9835:                      B      : TVector;
9836:                      V      : TMatrix); external 'dmath';
9837: { Polynomial regression by Gauss-Jordan method }
9838: procedure WPolFit(X, Y, S : TVector;
9839:                      Lb, Ub, Deg : Integer;
9840:                      B      : TVector;
9841:                      V      : TMatrix); external 'dmath';
9842: { Weighted polynomial regression by Gauss-Jordan method }
9843: procedure SVDPolFit(X, Y : TVector;
9844:                       Lb, Ub, Deg : Integer;
9845:                       SVDTol : Float;
9846:                       B      : TVector;
9847:                       V      : TMatrix); external 'dmath';
9848: { Unweighted polynomial regression by singular value decomposition }
9849: procedure WSVDPolFit(X, Y, S : TVector;
9850:                        Lb, Ub, Deg : Integer;
9851:                        SVDTol : Float;
9852:                        B      : TVector;
9853:                        V      : TMatrix); external 'dmath';
9854: { Weighted polynomial regression by singular value decomposition }
9855: procedure RegTest(Y, Ycalc : TVector;
9856:                      LbY, UbY : Integer;
9857:                      V      : TMatrix;
9858:                      LbV, UbV : Integer;
9859:                      var Test : TRegTest); external 'dmath';
9860: { Test of unweighted regression }
9861: procedure WRegTest(Y, Ycalc, S : TVector;
9862:                      LbY, UbY : Integer;
9863:                      V      : TMatrix;
9864:                      LbV, UbV : Integer;
9865:                      var Test : TRegTest); external 'dmath';
9866: { Test of weighted regression }
9867: { -----
9868: Nonlinear regression
9869: ----- }
9870: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9871: { Sets the optimization algorithm for nonlinear regression }
9872: function GetOptAlgo : TOptAlgo; external 'dmath';
9873: { Returns the optimization algorithm }
9874: procedure SetMaxParam(N : Byte); external 'dmath';
9875: { Sets the maximum number of regression parameters for nonlinear regression }
9876: function GetMaxParam : Byte; external 'dmath';

```

```

9877: { Returns the maximum number of regression parameters for nonlinear regression }
9878: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9879: { Sets the bounds on the I-th regression parameter }
9880: procedure GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
9881: { Returns the bounds on the I-th regression parameter }
9882: procedure NLFit(RegFunc : TRegFunc;
9883:                      DerivProc : TDerivProc;
9884:                      X, Y      : TVector;
9885:                      Lb, Ub    : Integer;
9886:                      MaxIter   : Integer;
9887:                      Tol       : Float;
9888:                      B          : TVector;
9889:                      FirstPar, LastPar : Integer;
9890:                      V          : TMatrix); external 'dmath';
9891: { Unweighted nonlinear regression }
9892: procedure WNLFit(RegFunc : TRegFunc;
9893:                      DerivProc : TDerivProc;
9894:                      X, Y, S   : TVector;
9895:                      Lb, Ub    : Integer;
9896:                      MaxIter   : Integer;
9897:                      Tol       : Float;
9898:                      B          : TVector;
9899:                      FirstPar, LastPar : Integer;
9900:                      V          : TMatrix); external 'dmath';
9901: { Weighted nonlinear regression }
9902: procedure SetMCFile(FileName : String); external 'dmath';
9903: { Set file for saving MCMC simulations }
9904: procedure SimFit(RegFunc : TRegFunc;
9905:                      X, Y      : TVector;
9906:                      Lb, Ub    : Integer;
9907:                      B          : TVector;
9908:                      FirstPar, LastPar : Integer;
9909:                      V          : TMatrix); external 'dmath';
9910: { Simulation of unweighted nonlinear regression by MCMC }
9911: procedure WSimFit(RegFunc : TRegFunc;
9912:                      X, Y, S   : TVector;
9913:                      Lb, Ub    : Integer;
9914:                      B          : TVector;
9915:                      FirstPar, LastPar : Integer;
9916:                      V          : TMatrix); external 'dmath';
9917: { Simulation of weighted nonlinear regression by MCMC }
9918: { -----
9919: Nonlinear regression models
9920: ----- }
9921: { -----
9922: -----
9923: -----
9924: -----
9925: procedure FracFit(X, Y      : TVector;
9926:                      Lb, Ub    : Integer;
9927:                      Degl, Deg2 : Integer;
9928:                      ConsTerm  : Boolean;
9929:                      MaxIter   : Integer;
9930:                      Tol       : Float;
9931:                      B          : TVector;
9932:                      V          : TMatrix); external 'dmath';
9933: { Unweighted fit of rational fraction }
9934: procedure WFracFit(X, Y, S   : TVector;
9935:                      Lb, Ub    : Integer;
9936:                      Degl, Deg2 : Integer;
9937:                      ConsTerm  : Boolean;
9938:                      MaxIter   : Integer;
9939:                      Tol       : Float;
9940:                      B          : TVector;
9941:                      V          : TMatrix); external 'dmath';
9942: { Weighted fit of rational fraction }
9943: { -----
9944: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9945: { Returns the value of the rational fraction at point X }
9946: procedure ExpFit(X, Y      : TVector;
9947:                      Lb, Ub, Nexp : Integer;
9948:                      ConsTerm  : Boolean;
9949:                      MaxIter   : Integer;
9950:                      Tol       : Float;
9951:                      B          : TVector;
9952:                      V          : TMatrix); external 'dmath';
9953: { Unweighted fit of sum of exponentials }
9954: procedure WExpFit(X, Y, S   : TVector;
9955:                      Lb, Ub, Nexp : Integer;
9956:                      ConsTerm  : Boolean;
9957:                      MaxIter   : Integer;
9958:                      Tol       : Float;
9959:                      B          : TVector;
9960:                      V          : TMatrix); external 'dmath';
9961: { Weighted fit of sum of exponentials }
9962: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9963: { Returns the value of the regression function at point X }
9964: procedure IncExpFit(X, Y      : TVector;
9965:                      Lb, Ub    : Integer;

```

```

9966:           ConsTerm : Boolean;
9967:           MaxIter  : Integer;
9968:           Tol      : Float;
9969:           B        : TVector;
9970:           V        : TMatrix); external 'dmath';
9971: { Unweighted fit of model of increasing exponential }
9972: procedure WIIncExpFit(X, Y, S : TVector;
9973:                           Lb, Ub : Integer;
9974:                           ConsTerm : Boolean;
9975:                           MaxIter  : Integer;
9976:                           Tol      : Float;
9977:                           B        : TVector;
9978:                           V        : TMatrix); external 'dmath';
9979: { Weighted fit of increasing exponential }
9980: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9981: { Returns the value of the regression function at point X }
9982: procedure ExpLinFit(X, Y : TVector;
9983:                         Lb, Ub : Integer;
9984:                         MaxIter : Integer;
9985:                         Tol      : Float;
9986:                         B        : TVector;
9987:                         V        : TMatrix); external 'dmath';
9988: { Unweighted fit of the "exponential + linear" model }
9989: procedure WExPLinFit(X, Y, S : TVector;
9990:                         Lb, Ub : Integer;
9991:                         MaxIter : Integer;
9992:                         Tol      : Float;
9993:                         B        : TVector;
9994:                         V        : TMatrix); external 'dmath';
9995: { Weighted fit of the "exponential + linear" model }
9996:
9997: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9998: { Returns the value of the regression function at point X }
9999: procedure MichFit(X, Y : TVector;
10000:                      Lb, Ub : Integer;
10001:                      MaxIter : Integer;
10002:                      Tol      : Float;
10003:                      B        : TVector;
10004:                      V        : TMatrix); external 'dmath';
10005: { Unweighted fit of Michaelis equation }
10006: procedure WMichFit(X, Y, S : TVector;
10007:                         Lb, Ub : Integer;
10008:                         MaxIter : Integer;
10009:                         Tol      : Float;
10010:                         B        : TVector;
10011:                         V        : TMatrix); external 'dmath';
10012: { Weighted fit of Michaelis equation }
10013: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10014: { Returns the value of the Michaelis equation at point X }
10015: procedure MintFit(X, Y : TVector;
10016:                      Lb, Ub : Integer;
10017:                      MintVar : TMintVar;
10018:                      Fit_S0  : Boolean;
10019:                      MaxIter : Integer;
10020:                      Tol      : Float;
10021:                      B        : TVector;
10022:                      V        : TMatrix); external 'dmath';
10023: { Unweighted fit of the integrated Michaelis equation }
10024: procedure WMintFit(X, Y, S : TVector;
10025:                         Lb, Ub : Integer;
10026:                         MintVar : TMintVar;
10027:                         Fit_S0  : Boolean;
10028:                         MaxIter : Integer;
10029:                         Tol      : Float;
10030:                         B        : TVector;
10031:                         V        : TMatrix); external 'dmath';
10032: { Weighted fit of the integrated Michaelis equation }
10033: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10034: { Returns the value of the integrated Michaelis equation at point X }
10035: procedure HillFit(X, Y : TVector;
10036:                      Lb, Ub : Integer;
10037:                      MaxIter : Integer;
10038:                      Tol      : Float;
10039:                      B        : TVector;
10040:                      V        : TMatrix); external 'dmath';
10041: { Unweighted fit of Hill equation }
10042: procedure WHillFit(X, Y, S : TVector;
10043:                         Lb, Ub : Integer;
10044:                         MaxIter : Integer;
10045:                         Tol      : Float;
10046:                         B        : TVector;
10047:                         V        : TMatrix); external 'dmath';
10048: { Weighted fit of Hill equation }
10049: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10050: { Returns the value of the Hill equation at point X }
10051: procedure LogiFit(X, Y : TVector;
10052:                         Lb, Ub : Integer;
10053:                         ConsTerm : Boolean;
10054:                         General : Boolean;

```

```

10055:           MaxIter : Integer;
10056:           Tol    : Float;
10057:           B      : TVector;
10058:           V      : TMatrix); external 'dmath';
10059: { Unweighted fit of logistic function }
10060: procedure WLogiFit(X, Y, S : TVector;
10061:                      Lb, Ub : Integer;
10062:                      ConstTerm : Boolean;
10063:                      General : Boolean;
10064:                      MaxIter : Integer;
10065:                      Tol    : Float;
10066:                      B      : TVector;
10067:                      V      : TMatrix); external 'dmath';
10068: { Weighted fit of logistic function }
10069: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10070: { Returns the value of the logistic function at point X }
10071: procedure PKFit(X, Y : TVector;
10072:                      Lb, Ub : Integer;
10073:                      MaxIter : Integer;
10074:                      Tol    : Float;
10075:                      B      : TVector;
10076:                      V      : TMatrix); external 'dmath';
10077: { Unweighted fit of the acid-base titration curve }
10078: procedure WPKFit(X, Y, S : TVector;
10079:                      Lb, Ub : Integer;
10080:                      MaxIter : Integer;
10081:                      Tol    : Float;
10082:                      B      : TVector;
10083:                      V      : TMatrix); external 'dmath';
10084: { Weighted fit of the acid-base titration curve }
10085: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10086: { Returns the value of the acid-base titration function at point X }
10087: procedure PowFit(X, Y : TVector;
10088:                      Lb, Ub : Integer;
10089:                      MaxIter : Integer;
10090:                      Tol    : Float;
10091:                      B      : TVector;
10092:                      V      : TMatrix); external 'dmath';
10093: { Unweighted fit of power function }
10094: procedure WPowFit(X, Y, S : TVector;
10095:                      Lb, Ub : Integer;
10096:                      MaxIter : Integer;
10097:                      Tol    : Float;
10098:                      B      : TVector;
10099:                      V      : TMatrix); external 'dmath';
10100: { Weighted fit of power function }
10101:
10102: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10103: { Returns the value of the power function at point X }
10104: procedure GammaFit(X, Y : TVector;
10105:                      Lb, Ub : Integer;
10106:                      MaxIter : Integer;
10107:                      Tol    : Float;
10108:                      B      : TVector;
10109:                      V      : TMatrix); external 'dmath';
10110: { Unweighted fit of gamma distribution function }
10111: procedure WGammaFit(X, Y, S : TVector;
10112:                      Lb, Ub : Integer;
10113:                      MaxIter : Integer;
10114:                      Tol    : Float;
10115:                      B      : TVector;
10116:                      V      : TMatrix); external 'dmath';
10117: { Weighted fit of gamma distribution function }
10118: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10119: { Returns the value of the gamma distribution function at point X }
10120: { -----
10121:   Principal component analysis
10122:   ----- }
10123: procedure VecMean(X : TMatrix;
10124:                      Lb, Ub, Nvar : Integer;
10125:                      M      : TVector); external 'dmath';
10126: { Computes the mean vector M from matrix X }
10127: procedure VecSD(X : TMatrix;
10128:                      Lb, Ub, Nvar : Integer;
10129:                      M, S      : TVector); external 'dmath';
10130: { Computes the vector of standard deviations S from matrix X }
10131: procedure MatVarCov(X : TMatrix;
10132:                      Lb, Ub, Nvar : Integer;
10133:                      M      : TVector;
10134:                      V      : TMatrix); external 'dmath';
10135: { Computes the variance-covariance matrix V from matrix X }
10136: procedure MatCorrel(V : TMatrix;
10137:                      Nvar : Integer;
10138:                      R      : TMatrix); external 'dmath';
10139: { Computes the correlation matrix R from the var-cov matrix V }
10140: procedure PCA(R : TMatrix;
10141:                      Nvar : Integer;
10142:                      Lambda : TVector;
10143:                      C, Rc : TMatrix); external 'dmath';

```

```

10144: { Performs a principal component analysis of the correlation matrix R }
10145: procedure ScaleVar(X : TMatrix;
10146:           Lb, Ub, Nvar : Integer;
10147:           M, S : TVector;
10148:           Z : TMATRIX); external 'dmath';
10149: { Scales a set of variables by subtracting means and dividing by SD's }
10150: procedure PrinFac(Z : TMatrix;
10151:           Lb, Ub, Nvar : Integer;
10152:           C, F : TMATRIX); external 'dmath';
10153: { Computes principal factors }
10154: { -----
10155:   Strings
10156:   ----- }
10157: function LTrim(S : String) : String; external 'dmath';
10158: { Removes leading blanks }
10159: function RTrim(S : String) : String; external 'dmath';
10160: { Removes trailing blanks }
10161: function Trim(S : String) : String; external 'dmath';
10162: { Removes leading and trailing blanks }
10163: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10164: { Returns a string made of character C repeated N times }
10165: function RFill(S : String; L : Byte) : String; external 'dmath';
10166: { Completes string S with trailing blanks for a total length L }
10167: function LFill(S : String; L : Byte) : String; external 'dmath';
10168: { Completes string S with leading blanks for a total length L }
10169: function CFill(S : String; L : Byte) : String; external 'dmath';
10170: { Centers string S on a total length L }
10171: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10172: { Replaces in string S all the occurrences of C1 by C2 }
10173: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10174: { Extracts a field from a string }
10175: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10176: { Parses a string into its constitutive fields }
10177: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10178: { Sets the numeric format }
10179: function FloatStr(X : Float) : String; external 'dmath';
10180: { Converts a real to a string according to the numeric format }
10181: function IntStr(N : LongInt) : String; external 'dmath';
10182: { Converts an integer to a string }
10183: function CompStr(Z : Complex) : String; external 'dmath';
10184: { Converts a complex number to a string }
10185: {$IFDEF DELPHI}
10186: function StrDec(S : String) : String; external 'dmath';
10187: { Set decimal separator to the symbol defined in SysUtils }
10188: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10189: { Test if a string represents a number and returns it in X }
10190: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10191: { Reads a floating point number from an Edit control }
10192: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10193: { Writes a floating point number in a text file }
10194: {$ENDIF}
10195: { -----
10196:   BGI / Delphi graphics
10197:   ----- }
10198: function InitGraphics
10199: {$IFDEF DELPHI}
10200: (Width, Height : Integer) : Boolean;
10201: {$ELSE}
10202: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10203: { Enters graphic mode }
10204: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10205:           X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10206: { Sets the graphic window }
10207: procedure SetOxScale(Scale : TScale;
10208:           OxMin, OxMax, OxStep : Float); external 'dmath';
10209: { Sets the scale on the Ox axis }
10210: procedure SetOyScale(Scale : TScale;
10211:           OyMin, OyMax, OyStep : Float); external 'dmath';
10212: { Sets the scale on the Oy axis }
10213: procedure GetOxScale(var Scale : TScale;
10214:           var OxMin, OxMax, OxStep : Float); external 'dmath';
10215: { Returns the scale on the Ox axis }
10216: procedure GetOyScale(var Scale : TScale;
10217:           var OyMin, OyMax, OyStep : Float); external 'dmath';
10218: { Returns the scale on the Oy axis }
10219: procedure SetGraphTitle>Title : String); external 'dmath'; { Sets the title for the graph }
10220: procedure SetOxTitle>Title : String); external 'dmath'; { Sets the title for the Ox axis }
10221: procedure SetOyTitle>Title : String); external 'dmath'; { Sets the title for the Oy axis }
10222: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10223: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10224: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10225: {$IFNDEF DELPHI}
10226: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10227: { Sets the font for the main graph title }
10228: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10229: { Sets the font for the Ox axis (title and labels) }
10230: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10231: { Sets the font for the Oy axis (title and labels) }
10232: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';

```

```

10233: { Sets the font for the legends }
10234: procedure SetClipping(Clip : Boolean); external 'dmath';
10235: { Determines whether drawings are clipped at the current viewport
10236:   boundaries, according to the value of the Boolean parameter Clip }
10237: {$ENDIF}
10238: procedure PlotOxAxis({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10239: { Plots the horizontal axis }
10240: procedure PlotOyAxis({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10241: { Plots the vertical axis }
10242: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10243: { Plots a grid on the graph }
10244: procedure WriteGraphTitle({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10245: { Writes the title of the graph }
10246: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10247: { Sets the maximum number of curves and re-initializes their parameters }
10248: procedure SetPointParam
10249: {$IFDEF DELPHI}
10250: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10251: {$ELSE}
10252: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10253: { Sets the point parameters for curve # CurvIndex }
10254: procedure SetlineParam
10255: {$IFDEF DELPHI}
10256: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10257: {$ELSE}
10258: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10259: { Sets the line parameters for curve # CurvIndex }
10260: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10261: { Sets the legend for curve # CurvIndex }
10262: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10263: { Sets the step for curve # CurvIndex }
10264: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10265: procedure GetPointParam
10266: {$IFDEF DELPHI}
10267: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10268: {$ELSE}
10269: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10270: { Returns the point parameters for curve # CurvIndex }
10271: procedure GetLineParam
10272: {$IFDEF DELPHI}
10273: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10274: {$ELSE}
10275: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10276: { Returns the line parameters for curve # CurvIndex }
10277: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10278: { Returns the legend for curve # CurvIndex }
10279: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10280: { Returns the step for curve # CurvIndex }
10281: {$IFDEF DELPHI}
10282: procedure PlotPoint(Canvas : TCanvas;
10283:                      X, Y : Float; CurvIndex : Integer); external 'dmath';
10284: {$ELSE}
10285: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10286: {$ENDIF}
10287: { Plots a point on the screen }
10288: procedure PlotCurve({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10289:                      X, Y : TVector;
10290:                      Lb, Ub, CurvIndex : Integer); external 'dmath';
10291: { Plots a curve }
10292: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10293:                                     X, Y, S : TVector;
10294:                                     Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10295: { Plots a curve with error bars }
10296: procedure PlotFunc({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10297:                      Func : TFunc;
10298:                      Xmin, Xmax : Float;
10299:                      {$IFDEF DELPHI}Npt : Integer;{$ENDIF}
10300:                      CurvIndex : Integer); external 'dmath';
10301: { Plots a function }
10302: procedure WriteLegend({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10303:                          NCurv : Integer;
10304:                          ShowPoints, ShowLines : Boolean); external 'dmath';
10305: { Writes the legends for the plotted curves }
10306: procedure ConRec({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10307:                      Nx, Ny, Nc : Integer;
10308:                      X, Y, Z : TVector;
10309:                      F : TMatrix); external 'dmath';
10310: { Contour plot }
10311: function Xpixel(X : Float):Integer; external 'dmath'; { Converts user abscissa X to screen coordinate }
10312: function Ypixel(Y : Float):Integer; external 'dmath'; { Converts user ordinate Y to screen coordinate }
10313: function Xuser(X : Integer):Float; external 'dmath'; { Converts screen coordinate X to user abscissa }
10314: function Yuser(Y : Integer):Float; external 'dmath'; { Converts screen coordinate Y to user ordinate }
10315: {$IFDEF DELPHI}
10316: procedure LeaveGraphics; external 'dmath';
10317: { Quits graphic mode }
10318: {$ENDIF}
10319: { -----
10320:  LaTeX graphics
10321:  ----- }

```

```

10322: function TeX_InitGraphics(FileName : String; PgWidth, PgHeight : Integer;
10323:                               Header : Boolean) : Boolean; external 'dmath';
10324: { Initializes the LaTeX file }
10325: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10326: { Sets the graphic window }
10327: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10328: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10329: { Sets the scale on the Ox axis }
10330: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10331: { Sets the scale on the Oy axis }
10332: procedure TeX_SetGraphTitle>Title : String); external 'dmath'; { Sets the title for the graph }
10333: procedure TeX_SetOxTitle>Title : String); external 'dmath'; { Sets the title for the Ox axis }
10334: procedure TeX_SetOyTitle>Title : String); external 'dmath'; { Sets the title for the Oy axis }
10335: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10336: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10337: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10338: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10339: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10340: { Sets the maximum number of curves and re-initializes their parameters }
10341: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10342: { Sets the point parameters for curve # CurvIndex }
10343: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10344:                               Width : Float; Smooth : Boolean); external 'dmath';
10345: { Sets the line parameters for curve # CurvIndex }
10346: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10347: { Sets the legend for curve # CurvIndex }
10348: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10349: { Sets the step for curve # CurvIndex }
10350: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10351: { Plots a curve }
10352: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10353:                                         Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10354: { Plots a curve with error bars }
10355: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10356:                           Npt : Integer; CurvIndex : Integer); external 'dmath';
10357: { Plots a function }
10358: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10359: { Writes the legends for the plotted curves }
10360: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix); external 'dmath';
10361: { Contour plot }
10362: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10363: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10364:
10365: //*****unit uPSI_SynPdf;
10366: Function RawUTF8ToPDFString( const Value : RawUTF8 ) : PDFString
10367: Function _DateTimeToPdfDate( ADate : TDateTime ) : TPdfDate
10368: Function _PdfToDateText( const AText : TPdfDate ) : TDateTime
10369: Function PdfRect( Left, Top, Right, Bottom : Single ) : TPdfRect;
10370: Function PdfRect1( const Box : TPdfBox ) : TPdfRect;
10371: Function PdfBox( Left, Top, Width, Height : Single ) : TPdfBox
10372: //Function _GetCharCount( Text : PWideChar ) : integer
10373: //Procedure L2R( W : PWideChar; L : integer )
10374: Function PdfCoord( MM : single ) : integer
10375: Function CurrentPrinterPageSize : TPDFPaperSize
10376: Function CurrentPrinterRes : TPoint
10377: Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8 )
10378: Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer )
10379: Procedure GDICommentLink( MetaHandle: HDC; const aBookmarkName:RawUTF8; const aRect : TRect )
10380: Const ('Usp10','String 'usp10.dll
10381: AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10382: 'fcharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10383: AddTypeS('TScriptState_set', 'set of TScriptState_enum
10384: //*****
10385:
10386: procedure SIRegister_PMrand(CL: TPSPascalCompiler); //ParkMiller
10387: begin
10388: Procedure PMrandomize( I : word)
10389: Function PMrandom : longint
10390: Function Rrand : extended
10391: Function Irand( N : word ) : word
10392: Function Brand( P : extended ) : boolean
10393: Function Nrand : extended
10394: end;
10395:
10396: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10397: begin
10398:   Function Endian( x : LongWord ) : LongWord
10399:   Function Endian64( x : Int64 ) : Int64
10400:   Function spRol( x : LongWord; y : Byte) : LongWord
10401:   Function spRor( x : LongWord; y : Byte) : LongWord
10402:   Function Ror64( x : Int64; y : Byte) : Int64
10403: end;
10404:
10405: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10406: begin
10407: Procedure ClearModules
10408: Procedure ReadMapfile( Fname : string )
10409: Function AddressInfo( Address : dword ) : string
10410: end;

```

```

10411:
10412: procedure SIRегистер_LibTar(CL: TPSPascalCompiler);
10413: begin
10414:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOwner
10415:     + 'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWriteByOther
10416:     + 'teByOther, tpExecuteByOther )'
10417:   TTarPermissions', 'set of TTarPermission
10418:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft
10419:     + 'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10420:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )'
10421:   TTarModes', 'set of TTarMode
10422:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDA
10423:     + 'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST
10424:     + 'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;
10425:     + 'ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE
10426:     + 'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10427:   SIRегистер_TTarArchive(CL);
10428:   SIRегистер_TTarWriter(CL);
10429:   Function PermissionString( Permissions : TTarPermissions ) : STRING
10430:   Function ConvertFilename( Filename : STRING ) : STRING
10431:   Function FileTimeGMT( FileName : STRING ) : TDateTime;
10432:   Function FileTimeGMT1( SearchRec : TSearchRec ) : TDateTime;
10433:   Procedure ClearDirRec( var DirRec : TTarDirRec )
10434: end;
10435:
10436:
10437: //*****unit uPSI_TlHelp32;
10438: procedure SIRегистер_TlHelp32(CL: TPSPascalCompiler);
10439: begin
10440:   Const('MAX_MODULE_NAME32','LongInt'( 255 );
10441:   Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD ) : THandle
10442:   Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001 );
10443:   Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002 );
10444:   Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004 );
10445:   Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008 );
10446:   Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000 );
10447:   tagHEAPLIST32', 'record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10448:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10449:   AddTypeS('THeapList32', 'tagHEAPLIST32
10450:   Const('HF32_DEFAULT','LongInt'( 1 );
10451:   Const('HF32_SHARED','LongInt'( 2 );
10452:   Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10453:   Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10454:   AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAdr
10455:     + 'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10456:     + 'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10457:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10458:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10459:   Const('LF32_FIXED','LongWord').SetUInt( $00000001 );
10460:   Const('LF32_FREE','LongWord').SetUInt( $00000002 );
10461:   Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004 );
10462:   Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD ) : BOOL
10463:   Function Heap32Next( var lphe : THeapEntry32 ) : BOOL
10464:   DWORD; var lpNumberOfBytesRead : DWORD ) : BOOL
10465:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3
10466:     + '2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt
10467:     + 'aPri : Longint; dwFlags : DWORD; end
10468:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10469:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10470:   Function Thread32First( hSnapshot : THandle; var lppte : TThreadEntry32 ) : BOOL
10471:   Function Thread32Next( hSnapshot : THandle; var lppte : TThreadEntry32 ) : BOOL
10472: end;
10473:   Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042 );
10474:   Const('EW_REBOOTSYSTEM','LongWord( $0043 );
10475:   Const('EW_EXITANDEXECAPP','LongWord( $0044 );
10476:   Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ) );
10477:   Const('EWX_LOGOFF','LongInt'( 0 );
10478:   Const('EWX_SHUTDOWN','LongInt'( 1 );
10479:   Const('EWX_REBOOT','LongInt'( 2 );
10480:   Const('EWX_FORCE','LongInt'( 4 );
10481:   Const('EWX_POWEROFF','LongInt'( 8 );
10482:   Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10 );
10483:   Function GET_APPCOMMAND_LPARAM( const lParam : LongInt ) : Shortint
10484:   Function GET_DEVICE_LPARAM( const lParam : LongInt ) : Word
10485:   Function GET_MOUSEKEY_LPARAM( const lParam : LongInt ) : Word
10486:   Function GET_FLAGS_LPARAM( const lParam : LongInt ) : Word
10487:   Function GET_KEYSTATE_LPARAM( const lParam : LongInt ) : Word
10488:   Function GetWindowWord( hWnd : HWND; nIndex : Integer ) : Word
10489:   Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word ) : Word
10490:   Function GetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
10491:   Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint
10492:   Function GetClassWord( hWnd : HWND; nIndex : Integer ) : Word
10493:   Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word ) : Word
10494:   Function GetClassLong( hWnd : HWND; nIndex : Integer ) : DWORD
10495:   Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
10496:   Function GetDesktopWindow : HWND
10497:   Function GetParent( hWnd : HWND ) : HWND
10498:   Function SetParent( hWndChild, hWndNewParent : HWND ) : HWND
10499:   Function GetTopWindow( hWnd : HWND ) : HWND

```

```

10500: Function GetNextWindow( hWnd : HWND; uCmd : UINT ) : HWND
10501: Function GetWindow( hWnd : HWND; uCmd : UINT ) : HWND
10502: //Delphi DFM
10503: Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10504: Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string ) : integer
10505: procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10506: function GetHighlightersFilter(AHighlighters: TStringList): string;
10507: function GetHighlighterFromfileExt(AHighlighters: TStringList; Extension: string):TSynCustomHighlighter;
10508: Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL ) : BOOL
10509: Function OpenIcon( hWnd : HWND ) : BOOL
10510: Function CloseWindow( hWnd : HWND ) : BOOL
10511: Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL ) : BOOL
10512: Function SetWindowPos(hWnd : HWND; hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UInt) : BOOL
10513: Function IsWindowVisible( hWnd : HWND ) : BOOL
10514: Function IsIconic( hWnd : HWND ) : BOOL
10515: Function AnyPopup : BOOL
10516: Function BringWindowToTop( hWnd : HWND ) : BOOL
10517: Function IsZoomed( hWnd : HWND ) : BOOL
10518: Function IsWindow( hWnd : HWND ) : BOOL
10519: Function IsMenu( hMenu : HMENU ) : BOOL
10520: Function IsChild( hWndParent, hWnd : HWND ) : BOOL
10521: Function DestroyWindow( hWnd : HWND ) : BOOL
10522: Function ShowWindow( hWnd : HWND; nCmdShow : Integer ) : BOOL
10523: Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD ) : BOOL
10524: Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer ) : BOOL
10525: Function FlashWindow( hWnd : HWND; bInvert : BOOL ) : BOOL
10526: Function IsWindowUnicode( hWnd : HWND ) : BOOL
10527: Function EnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL
10528: Function IsWindowEnabled( hWnd : HWND ) : BOOL
10529:
10530: procedure SIRегистre_IDECmdLine(CL: TPSPascalCompiler);
10531: begin
10532:   const ('ShowSetupDialogOptLong', 'String '--setup
10533: PrimaryConfPathOptLong', 'String '--primary-config-path=
10534: PrimaryConfPathOptShort', 'String '--pcp=
10535: SecondaryConfPathOptLong', 'String '--secondary-config-path=
10536: SecondaryConfPathOptShort', 'String '--scp=
10537: NoSplashScreenOptLong', 'String '--no-splash-screen
10538: NoSplashScreenOptShort', 'String '--nsc
10539: StartedByStartLazarusOpt', 'String '--started-by-startlazarus
10540: SkipLastProjectOpt', 'String '--skip-last-project
10541: DebugLogOpt', 'String '--debug-log=
10542: DebugLogOptEnable', 'String '--debug-enable=
10543: LanguageOpt', 'String '--language=
10544: LazarusDirOpt', 'String '--lazarusdir=
10545: Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10546: Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10547: Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings ) : string
10548: Function IsHelpRequested : Boolean
10549: Function IsVersionRequested : boolean
10550: Function GetLanguageSpecified : string
10551: Function ParamIsOption( ParamIndex : integer; const Option : string ) : boolean
10552: Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10553: Procedure ParseNoGuiCmdLineParams
10554: Function ExtractCmdLineFilenames : TStrings
10555: end;
10556:
10557:
10558: procedure SIRегистre_LazFileUtils(CL: TPSPascalCompiler);
10559: begin
1060:   Function CompareFilenames( const Filenam1, Filenam2 : string ) : integer
1061:   Function CompareFilenamesIgnoreCase( const Filenam1, Filenam2 : string ) : integer
1062:   Function CompareFileExt( const Filenam, Ext : string; CaseSensitive : boolean ) : integer;
1063:   Function CompareFileExt1( const Filenam, Ext : string ) : integer;
1064:   Function CompareFilenameStarts( const Filenam1, Filenam2 : string ) : integer
1065:   Function CompareFilenames(Filenam1:PChar;Len1:integer; Filenam2:PChar;Len2:integer):integer
1066:   Function CompareFilenamesP( Filenam1:PChar, Filenam2:PChar; IgnoreCase : boolean ) : integer
1067:   Function DirPathExists( DirectoryName : string ) : boolean
1068:   Function DirectoryIsWritable( const DirectoryName : string ) : boolean
1069:   Function ExtractFileNameOnly( const AFilename : string ) : string
1070:   Function FilenamIsAbsolute( const Thefilename : string ) : boolean
1071:   Function FilenamIsWinAbsolute( const Thefilename : string ) : boolean
1072:   Function FilenamIsUnixAbsolute( const Thefilename : string ) : boolean
1073:   Function ForceDirectory( DirectoryName : string ) : boolean
1074:   Procedure CheckIffileIsExecutable( const Afilename : string )
1075:   Procedure CheckIffileIsSymlink( const Afilename : string )
1076:   Function FileIsText( const Afilename : string ) : boolean
1077:   Function FileIsText2( const Afilename : string; out FileReadable : boolean ) : boolean
1078:   Function FilenamIsTrimmed( const Thefilename : string ) : boolean
1079:   Function FilenamIsTrimmed2( StartPos : PChar; NameLen : integer ) : boolean
1080:   Function TrimFilename( const Afilename : string ) : string
1081:   Function ResolveDots( const Afilename : string ) : string
1082:   Procedure ForcePathDelims( var FileName : string )
1083:   Function GetForcedPathDelims( const FileName : string ) : String
1084:   Function CleanAndExpandfilename( const Filename : string ) : string
1085:   Function CleanAndExpandDirectory( const Filename : string ) : string
1086:   Function TrimAndExpandfilename( const Filename : string; const BaseDir : string ) : string
1087:   Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string ) : string
1088:   Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
  AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String ) : Boolean

```

```

10589: Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
10590:   AlwaysRequireSharedBaseFolder: Boolean) : string
10591: Function FileIsInPath( const Filename, Path : string) : boolean
10592: Function AppendPathDelim( const Path : string) : string
10593: Function ChompPathDelim( const Path : string) : string
10594: Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10595: Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10596: Function MinimizeSearchPath( const SearchPath : string) : string
10597: Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10598: (*Function FileExistsUTF8( const FileName : string) : boolean
10599: Function FileAgeUTF8( const FileName : string) : Longint
10600: Function DirectoryExistsUTF8( const Directory : string) : Boolean
10601: Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10602: Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10603: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10604: Procedure FindCloseUTF8( var F : TSearchrec)
10605: Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10606: Function FileGetAttrUTF8( const FileName : String) : Longint
10607: Function DeleteFileUTF8( const FileName : String) : Boolean
10608: Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10609: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10610: Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10611: Function GetCurrentDirUTF8 : String
10612: Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10613: Function CreateDirUTF8( const NewDir : String) : Boolean
10614: Function RemoveDirUTF8( const Dir : String) : Boolean
10615: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10616: Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10617: Function FileCreateUTF8( const FileName : string) : THandle;
10618: Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10619: Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10620: Function FileSizeUTF8( const Filename : string) : int64
10621: Function GetFileDescription( const Afilename : string) : string
10622: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10623: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10624: Function GetTempFileNameUTF8( const Dir, Prefix : String*) : String*)
10625: Function IsUNCPath( const Path : String) : Boolean
10626: Function ExtractUNCVolume( const Path : String) : String
10627: Function ExtractFileRoot( FileName : String) : String
10628: Function GetDarwinSystemFilename( Filename : string) : string
10629: Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10630: Function StrToCmdLineParam( const Param : string) : string
10631: Function MergeCmdLineParams( ParamList : TStrings) : string
10632: Procedure InvalidateFileStateCache( const Filename : string)
10633: Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList;
10634: Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10635: Function FindAllDocs(const Root, extmask: string): TStringlist;
10636: Function ReadFileToString( const Filename : string) : string
10637: procedure Incl(var X: longint; N: Longint);
10638:
10639: type
10640:   TCopyFileFlag = ( cffOverwriteFile,
10641:     cffCreateDestDirectory, cffPreserveTime );
10642:   TCopyFileFlags = set of TCopyFileFlag;*)
10643:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10644:   TCopyFileFlags', 'set of TCopyFileFlag
10645:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10646: end;
10647:
10648: procedure SIRegister_lazMasks(CL: TPPascalCompiler);
10649: begin
10650:   TMaskCharType', '( mcChar, mc CharSet, mcAnyChar, mcAnyText )
10651:   SIRegister_TMask(CL);
10652:   SIRegister_TParseStringList(CL);
10653:   SIRegister_TMaskList(CL);
10654:   Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10655:   Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10656:   Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10657:   Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10658: end;
10659:
10660: procedure SIRegister_JvShellHook(CL: TPPascalCompiler);
10661: begin
10662:   //PShellHookInfo', '^TShellHookInfo // will not work
10663:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10664:   SHELLHOOKINFO', 'TShellHookInfo
10665:   LPSHELLHOOKINFO', 'PShellHookInfo
10666:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10667:   SIRegister_TJvShellHook(CL);
10668:   Function InitJvShellHooks : Boolean
10669:   Procedure UnInitJvShellHooks
10670: end;
10671:
10672: procedure SIRegister_JvExControls(CL: TPPascalCompiler);
10673: begin
10674:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10675:   +', dcHasSelSel, dcWantTab, dcNative )
10676:   TDlgCodes', 'set of TDlgCode

```

```

10677: 'dcWantMessage',' dcWantAllKeys);
10678: SIRegister_IJvExControl(CL);
10679: SIRegister_IJvDenySubClassing(CL);
10680: SIRegister_TStructPtrMessage(CL);
10681: Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10682: Procedure DrawDotNetBar( Control : TWinControl; AColor : TColor; InControl : Boolean);
10683: Procedure DrawDotNetBar1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10684: Procedure HandleDotNetBarHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10685: Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint ) : TMessage;
10686: Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl ) : TMessage;
10687: Function SmallPointToLong( const Pt : TSmallPoint ) : Longint
10688: Function ShiftStateToKeyData( Shift : TShiftState ) : Longint
10689: Function GetFocusedControl( AControl : TControl ) : TWInControl
10690: Function DlgcToDlgCodes( Value : Longint ) : TDlgCodes
10691: Function DlgCodesToDlcg( Value : TDlgCodes ) : Longint
10692: Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10693: Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage ) : Boolean
10694: SIRegister_TJvExControl(CL);
10695: SIRegister_TJvExWinControl(CL);
10696: SIRegister_TJvExCustomControl(CL);
10697: SIRegister_TJvExGraphicControl(CL);
10698: SIRegister_TJvExHintWindow(CL);
10699: SIRegister_TJvExPubGraphicControl(CL);
10700: end;
10701: (*-----*)
10702: procedure SIRegister_EncdDecd(CL: TPSPPascalCompiler);
10703: begin
10704:   Procedure EncodeStream( Input, Output : TStream)
10705:   Procedure DecodeStream( Input, Output : TStream)
10706:   Function EncodeString1( const Input : string) : string
10707:   Function DecodeString1( const Input : string) : string
10708: end;
10709: (*-----*)
10710: procedure SIRegister_SockAppReg(CL: TPSPPascalCompiler);
10711: begin
10712:   SIRegister_TWebAppRegInfo(CL);
10713:   SIRegister_TWebAppRegList(CL);
10714:   Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10715:   Procedure RegisterWebApp( const AFileName, AProgID : string)
10716:   Procedure UnregisterWebApp( const AProgID : string)
10717:   Function FindRegisteredWebApp( const AProgID : string) : string
10718:   Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10719:   'sUDPPort','String 'UDPPort
10720: end;
10721: (*-----*)
10722: procedure SIRegister_PJEnvVars(CL: TPSPPascalCompiler);
10723: begin
10724:   // TStringDynArray, 'array of string
10725:   Function GetEnvVarValue( const VarName : string) : string
10726:   Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10727:   Function DeleteEnvVar( const VarName : string) : Integer
10728:   Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
10729: BufSize:Int):Int;
10730:   Function ExpandEnvVars( const Str : string) : string
10731:   Function GetAllEnvVars( const Vars : TStrings) : Integer
10732:   Procedure GetAllEnvVarNames( const Names : TStrings);
10733:   Function GetAllEnvVarNames1 : TStringDynArray;
10734:   Function EnvBlockSize : Integer
10735:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10736:   SIRegister_TPJEnvVarsEnumerator(CL);
10737:   SIRegister_TPJEnvVars(CL);
10738:   FindClass('TOBJECT'), 'EPJEnvVars
10739:   FindClass('TOBJECT'), 'EPJEnvVars
10740:   //Procedure Register
10741: end;
10742: (*-----*)
10743: procedure SIRegister_PJConsoleApp(CL: TPSPPascalCompiler);
10744: begin
10745:   //cOneSecInMS', 'LongInt'( 1000);
10746:   //cDefTimeSlice', 'LongInt'( 50);
10747:   //cDefMaxExecTime', ' cOneMinInMS';
10748:   'cAppErrorMask', 'LongInt'( 1 shl 29);
10749:   Function IsApplicationError( const ErrCode : LongWord) : Boolean
10750:   TPJConsolePriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10751:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10752:   Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10753:   Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10754:   Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10755:   Function MakeSize( const ACX, ACY : LongInt) : TSize
10756:   SIRegister_TPJCustomConsoleApp(CL);
10757:   SIRegister_TPJConsoleApp(CL);
10758: end;
10759: (*-----*)
10760: procedure SIRegister_ip_misc(CL: TPSPPascalCompiler);
10761: begin
10762:   INVALID_IP_ADDRESS', 'LongWord').SetUInt( $ffffffff );
10763: end;

```

```

10765: t_encoding', '( uuencode, base64, mime )
10766: Function internet_date( date : TDateTime ) : string
10767: Function lookup_hostname( const hostname : string ) : longint
10768: Function my_hostname : string
10769: Function my_ip_address : longint
10770: Function ip2string( ip_address : longint ) : string
10771: Function resolve_hostname( ip : longint ) : string
10772: Function address_from( const s : string; count : integer ) : string
10773: Function encode_base64( data : TStream ) : TStringList
10774: Function decode_base64( source : TStringList ) : TMemoryStream
10775: Function posn( const s, t : string; count : integer ) : integer
10776: Function poscn( c : char; const s : string; n : integer ) : integer
10777: Function filename_of( const s : string ) : string
10778: //Function trim( const s : string ) : string
10779: //Procedure setlength( var s : string; l : byte)
10780: Function TimeZoneBias : longint
10781: Function eight2seven_quoteprint( const s : string ) : string
10782: Function eight2seven_german( const s : string ) : string
10783: Function seven2eight_quoteprint( const s : string ) : string end;
10784: type in_addr', 'record s_bytes : array[1..4] of byte; end;
10785: Function socketerror : cint
10786: Function fpsocket( domain : cint; xtype : cint; protocol : cint ) : cint
10787: Function frecv( s : cint; buf : __pointer; len : size_t; flags : cint ) : ssize_t
10788: Function fsend( s : cint; msg : __pointer; len : size_t; flags : cint ) : ssize_t
10789: //Function fbind( s : cint; addrx : psockaddr; addrlen : tsocklen ) : cint
10790: Function fplisten( s : cint; backlog : cint ) : cint
10791: //Function faccept( s : cint; addrx : psockaddr; addrlen : plongint ) : cint
10792: //Function fconnect( s : cint; name : psockaddr; namelen : tsocklen ) : cint
10793: //Function fgetsockname( s : cint; name : psockaddr; namelen : psocklen ) : cint
10794: Function NetAddrToStr( Entry : in_addr ) : String
10795: Function HostAddrToStr( Entry : in_addr ) : String
10796: Function StrToHostAddr( IP : String ) : in_addr
10797: Function StrToNetAddr( IP : String ) : in_addr
10798: SOL_SOCKET', 'LongWord').SetUInt( $ffff );
10799: cint8', 'shortint
10800: cuint8', 'byte
10801: cchar', 'cint8
10802: cschar', 'cint8
10803: uchar', 'cuint8
10804: cint16', 'smallint
10805: cuint16', 'word
10806: cshort', 'cint16
10807: csshort', 'cint16
10808: cushort', 'cuint16
10809: cint32', 'longint
10810: cuint32', 'longword
10811: cint', 'cint32
10812: csint', 'cint32
10813: cuint', 'cuint32
10814: csigned', 'cint
10815: cunsigned', 'cuint
10816: cint64', 'int64
10817: clonglong', 'cint64
10818: cslonglong', 'cint64
10819: cbool', 'longbool
10820: cfloat', 'single
10821: cdouble', 'double
10822: clongdouble', 'extended
10823:
10824: procedure SIRegister_uLkJSON(CL: TPPSPascalCompiler);
10825: begin
10826:   TlkJSONtypes', '( jsBase, jsNumber, jsString, jsBoolean, jsNull, jsList, jsObject )
10827:   SIRegister_TlkJSONdotnetclass(CL);
10828:   SIRegister_TlkJSONbase(CL);
10829:   SIRegister_TlkJSONnumber(CL);
10830:   SIRegister_TlkJSONstring(CL);
10831:   SIRegister_TlkJSONboolean(CL);
10832:   SIRegister_TlkJSONnull(CL);
10833:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Ele : TlkJSONba'
10834:     +'se; data : TObject; var Continue : Boolean)
10835:   SIRegister_TlkJSONcustomlist(CL);
10836:   SIRegister_TlkJSONlist(CL);
10837:   SIRegister_TlkJSONobjectmethod(CL);
10838:   TlkHashitem', 'record hash : cardinal; index : Integer; end
10839:   TlkHashFunction', 'Function ( const ws : WideString ) : cardinal
10840:   SIRegister_TlkHashTable(CL);
10841:   SIRegister_TlkBalTree(CL);
10842:   SIRegister_TlkJSONobject(CL);
10843:   SIRegister_TlkJSON(CL);
10844:   SIRegister_TlkJSONstreamed(CL);
10845:   Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10846: end;
10847:
10848: procedure SIRegister_ZSysUtils(CL: TPPSPascalCompiler);
10849: begin
10850:   TZListSortCompare', 'Function ( Item1, Item2 : TObject): Integer
10851:   SIRegister_TZSortedlist(CL);
10852:   Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10853:   Function zLastDelimiter( const Delimiters, Str : string) : Integer

```

```

10854: //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10855: //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10856: Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10857: Function StartsWithl( const Str, SubStr : RawByteString) : Boolean;
10858: Function EndsWithl( const Str, SubStr : WideString) : Boolean;
10859: Function EndsWithl( const Str, SubStr : RawByteString) : Boolean;
10860: Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10861: Function SQLStrToFloatDef( Str : String; Def : Extended) : Extended;
10862: Function SQLStrToFloat( const Str : AnsiString) : Extended;
10863: //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10864: //Function BufferToStr1( Buffer : PAbsiChar; Length : LongInt) : string;
10865: Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10866: Function StrToBoolEx( Str : string) : Boolean
10867: Function BoolToStrEx( Bool : Boolean) : String
10868: Function IsIpAddr( const Str : string) : Boolean
10869: Function zSplitString( const Str, Delimiters : string) : TStrings
10870: Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10871: Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10872: Function ComposeString( List : TStrings; const Delimiter : string) : string
10873: Function FloatToSQLStr( Value : Extended) : string
10874: Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10875: Function SplitStringEx( const Str, Delimiter : string) : TStrings
10876: Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10877: Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10878: Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10879: Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10880: Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10881: Function StrToBytes3( const Value : WideString) : TByteDynArray;
10882: Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10883: Function BytesToVar( const Value : TByteDynArray) : Variant
10884: Function VarToBytes( const Value : Variant) : TByteDynArray
10885: Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10886: Function TimestampStrToDateTime( const Value : string) : TDateTime
10887: Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10888: Function EncodeCString( const Value : string) : string
10889: Function DecodeCString( const Value : string) : string
10890: Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10891: Function MemPas( Buffer : PChar; Length : LongInt) : string
10892: Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
SubVersion:Int);
10893: Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
SubVersion:Integer):Int;
10894: Function FormatSQLVersion( const SQLVersion : Integer) : String
10895: Function ZStrToFloat( Value : AnsiChar) : Extended;
10896: Function ZStrToFloat1( Value : AnsiString) : Extended;
10897: Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10898: Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10899: Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10900: Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10901: Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10902: Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10903: Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10904: end;
10905:
10906: unit uPSI_ZEncoding;
10907: Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10908: Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10909: Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10910: Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10911: Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10912: Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10913: Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10914: Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10915: Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10916: Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10917: Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10918: Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10919: Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10920: Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10921: Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10922: Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10923: Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10924: Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10925: Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10926: Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10927: Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10928: Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10929: Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10930: Function ZConvertStringToUnicodeWithAutoEncode( const Src : String; const StringCP:Word):WideString
10931: Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10932: Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10933: Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10934: Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10935: Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10936: Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10937: Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10938: Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10939: Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10940: Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString

```

```

10941: Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word ) : String
10942: Function ZMoveStringToUTF8( const Src : String; const StringCP : Word ) : UTF8String
10943: Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word ) : WideString
10944: Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word ) : WideString
10945: Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word ) : RawByteString
10946: Function ZDefaultSystemCodePage : Word
10947: Function ZCompatibleCodePages( const CP1, CP2 : Word ) : Boolean
10948:
10949:
10950: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10951: begin
10952:   'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0 );
10953:   ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1 );
10954:   ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2 );
10955:   ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3 );
10956:   ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4 );
10957:   ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5 );
10958:   ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6 );
10959:   {('alDefault',1 RPC_C_AUTHN_LEVEL_DEFAULT),
10960:    ('alNone',2 RPC_C_AUTHN_LEVEL_NONE),
10961:    ('alConnect',3 RPC_C_AUTHN_LEVEL_CONNECT),
10962:    ('alCall',4 RPC_C_AUTHN_LEVEL_CALL),
10963:    ('alPacket',5 RPC_C_AUTHN_LEVEL_PKT),
10964:    ('alPacketIntegrity',6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY),
10965:    ('alPacketPrivacy',7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10966:   ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0 );
10967:   ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1 );
10968:   ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2 );
10969:   ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3 );
10970:   ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4 );
10971:   {('ilDefault',0 RPC_C_IMP_LEVEL_DEFAULT),
10972:    ('ilAnonymous',1 RPC_C_IMP_LEVEL_ANONYMOUS),
10973:    ('ilIdentiry',2 RPC_C_IMP_LEVEL_IDENTIFY),
10974:    ('ilImpersonate',3 RPC_C_IMP_LEVEL_IMPERSONATE),
10975:    ('ilDelegate',4 RPC_C_IMP_LEVEL_DELEGATE);}
10976:   ('EOAC_NONE','LongWord').SetUInt( $0 );
10977:   ('EOAC_DEFAULT','LongWord').SetUInt( $800 );
10978:   ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1 );
10979:   ('EOAC_STATIC_CLOAKING','LongWord').SetUInt( $20 );
10980:   ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40 );
10981:   ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80 );
10982:   ('RPC_C_AUTHN_WINNT','LongInt'( 10 );
10983:   ('RPC_C_AUTHNZ_NONE','LongInt'( 0 );
10984:   ('RPC_C_AUTHNZ_NAME','LongInt'( 1 );
10985:   ('RPC_C_AUTHNZ_DCE','LongInt'( 2 );
10986:   FindClass('TOBJECT'),'EBoldCom
10987: Function BoldVariantIsType( V : OleVariant; TypeCode : Integer ) : Boolean
10988: Function BoldMemoryToVariant( const Buffer, BufSize : Integer ) : OleVariant
10989: Function BoldStreamToVariant( Stream : TStream ) : OleVariant
10990: Function BoldStringsToVariant( Strings : TStrings ) : OleVariant
10991: Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer ) : Integer
10992: Function BoldVariantToStream( V : OleVariant; Stream : TStream ) : Integer
10993: Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings ) : Integer
10994: Function BoldVariantIsNamedValues( V : OleVariant ) : Boolean
10995: Function BoldCreateNamedValues( const Names:array of string;const Values:array of OleVariant):OleVariant;
10996: Function BoldGetNamedValue( Data : OleVariant; const Name : string ) : OleVariant
10997: Procedure BoldSetValue( Data : OleVariant; const Name : string; Value : OleVariant )
10998: Function BoldCreateGUID : TGUID
10999: Function BoldCreateComObject( const ClsId, IID : TGUID; out Obj : variant; out Res : HResult ) : Boolean
11000: Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out Res:HRes):Bool;
11001: Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint )
11002: Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown );
11003: end;
11004:
11005: (*-----*)
11006: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11007: begin
11008:   Function ParseISODate( s : string ) : TDateTime
11009:   Function ParseISODateTime( s : string ) : TDateTime
11010:   Function ParseISOTime( str : string ) : TDateTime
11011: end;
11012:
11013: (*-----*)
11014: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11015: begin
11016:   Function BoldCreateGUIDAsString( StripBrackets : Boolean ) : string
11017:   Function BoldCreateGUIDWithBracketsAsString : string
11018: end;
11019:
11020: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
11021: begin
11022:   FindClass('TOBJECT'),'TBoldFileHandler
11023:   FindClass('TOBJECT'),'TBoldDiskFileHandler
11024:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11025:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList )
11026:   SIRegister_TBoldFileHandler(CL);
11027:   SIRegister_TBoldDiskFileHandler(CL);
11028:   Procedure BoldCloseAllFilehandlers

```

```

11029: Procedure BoldRemoveUnchangedFilesFromEditor
11030: Function BoldFileHandlerList : TBoldObjectArray
11031: Function BoldFileHandlerForFile(path:FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
11032: OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
11033: end;
11034: procedure SIRегистер_BoldWinINet(CL: TPSPascalCompiler);
11035: begin
11036:   PCharArr', 'array of PChar
11037:   Function BoldInternetOpen(Agent:String;
11038:     AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr;
11039:   Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11040:   Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card):var
11041:   NumberOfBytesRead:Card):LongBool;
11042:   Function BoldInternetCloseHandle( HINet : Pointer ) : LongBool
11043:   Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
11044:   Cardinal; Reserved : Cardinal ) : LongBool
11045:   Function BoldInternetErrorDlg(hWnd:HWND:hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11046:   Function BoldInternetAttemptConnect( dwReserved : DWORD ) : DWORD
11047:   Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
11048:   Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11049:   Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11050: end;
11051: procedure SIRегистер_BoldQueryUserDlg(CL: TPSPascalCompiler);
11052: begin
11053:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11054:   SIRегистер_TfrmBoldQueryUser(CL);
11055:   Function QueryUser( const Title, Query : string ) : TBoldQueryResult
11056: end;
11057:
11058: (*-----*)
11059: procedure SIRегистер_BoldQueue(CL: TPSPascalCompiler);
11060: begin
11061:   //('befIsInDisplayList',' BoldElementFlag0 );
11062:   //('befStronglyDependedOfPrioritized',' BoldElementFlag1 );
11063:   //('befFollowerSelected',' BoldElementFlag2 );
11064:   FindClass('TOBJECT'), 'TBoldQueue
11065:   FindClass('TOBJECT'), 'TBoldQueueable
11066:   TBoldQueueDisplayStyle', '( dmDisplayOne, dmDisplayAll )
11067:   SIRегистер_TBoldQueueable(CL);
11068:   SIRегистер_TBoldQueue(CL);
11069:   Function BoldQueueFinalized : Boolean
11070:   Function BoldinstalledQueue : TBoldQueue
11071: end;
11072:
11073: procedure SIRегистер_BarcodE(CL: TPSPascalCompiler);
11074: begin
11075:   const mmPerInch','Extended').setExtended( 25.4);
11076:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,
11077:   +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11078:   +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11079:   +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11080:   +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11081:   TBarLineType', '( white, black, black_half )
11082:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11083:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st
11084:   +'pBottomLeft, stpBottomRight, stpBottomCenter )
11085:   TCheckSumMethod', '( csmNone, csmModulo10 )
11086:   SIRегистер_TAsBarcode(CL);
11087:   Function CheckSumModulo10( const data : string ) : string
11088:   Function ConvertMmToPixelsX( const Value : Double ) : Integer
11089:   Function ConvertMmToPixelsY( const Value : Double ) : Integer
11090:   Function ConvertInchToPixelsX( const Value : Double ) : Integer
11091:   Function ConvertInchToPixelsY( const Value : Double ) : Integer
11092: end;
11093:
11094: procedure SIRегистер_Geometry(CL: TPSPascalCompiler); //OpenGL
11095: begin
11096:   THomogeneousByteVector', 'array[0..3] of Byte
11097:   THomogeneousWordVector', 'array[0..3] of Word
11098:   THomogeneousIntVector', 'array[0..3] of Integer
11099:   THomogeneousFltVector', 'array[0..3] of single
11100:   THomogeneousDblVector', 'array[0..3] of double
11101:   THomogeneousExtVector', 'array[0..3] of extended
11102:   TAffineByteVector', 'array[0..2] of Byte
11103:   TAffineWordVector', 'array[0..2] of Word
11104:   TAffineIntVector', 'array[0..2] of Integer
11105:   TAffineFltVector', 'array[0..2] of single
11106:   TAffineDblVector', 'array[0..2] of double
11107:   TAffineExtVector', 'array[0..2] of extended
11108:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11109:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11110:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector

```

```

11111: THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11112: THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11113: THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11114: TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11115: TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11116: TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11117: TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11118: TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11119: TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11120: TMatrix4b', 'THomogeneousByteMatrix
11121: TMatrix4w', 'THomogeneousWordMatrix
11122: TMatrix4i', 'THomogeneousIntMatrix
11123: TMatrix4f', 'THomogeneousFltMatrix
11124: TMatrix4d', 'THomogeneousDblMatrix
11125: TMatrix4e', 'THomogeneousExtMatrix
11126: TMatrix3b', 'TAffineByteMatrix
11127: TMatrix3w', 'TAffineWordMatrix
11128: TMatrix3i', 'TAffineIntMatrix
11129: TMatrix3f', 'TAffineFltMatrix
11130: TMatrix3d', 'TAffineDblMatrix
11131: TMatrix3e', 'TAffineExtMatrix
11132: //''PMatrix', '^TMatrix // will not work
11133: TMatrixGL', 'THomogeneousFltMatrix
11134: THomogeneousMatrix', 'THomogeneousFltMatrix
11135: TAffineMatrix', 'TAffineFltMatrix
11136: TQuaternion', 'record Vector : TVector4f; end
11137: TRectangle', 'record Left : integer; Top : integer; Width : integer;
11138: +'ger; Height : Integer; end
11139: TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear
11140: +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11141: +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11142: 'EPSILON', 'Extended').setExtended( 1E-100 );
11143: 'EPSILON2', 'Extended').setExtended( 1E-50 );
11144: Function VectorAddGL( V1, V2 : TVectorGL ) : TVectorGL
11145: Function VectorAffineAdd( V1, V2 : TAffineVector ) : TAffineVector
11146: Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11147: Function VectorAffineDotProduct( V1, V2 : TAffineVector ) : Single
11148: Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single ) : TAffineVector
11149: Function VectorAffineSubtract( V1, V2 : TAffineVector ) : TAffineVector
11150: Function VectorAngle( V1, V2 : TAffineVector ) : Single
11151: Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single ) : TVectorGL
11152: Function VectorCrossProduct( V1, V2 : TAffineVector ) : TAffineVector
11153: Function VectorDotProduct( V1, V2 : TVectorGL ) : Single
11154: Function VectorLength( V : array of Single ) : Single
11155: Function VectorLerp( V1, V2 : TVectorGL; t : Single ) : TVectorGL
11156: Procedure VectorNegate( V : array of Single )
11157: Function VectorNorm( V : array of Single ) : Single
11158: Function VectorNormalize( V : array of Single ) : Single
11159: Function VectorPerpendicular( V, N : TAffineVector ) : TAffineVector
11160: Function VectorReflect( V, N : TAffineVector ) : TAffineVector
11161: Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single )
11162: Procedure VectorScale( V : array of Single; Factor : Single )
11163: Function VectorSubtractGL( V1, V2 : TVectorGL ) : TVectorGL
11164: Function CreateRotationMatrixX( Sine, Cosine : Single ) : TMatrixGL
11165: Function CreateRotationMatrixY( Sine, Cosine : Single ) : TMatrixGL
11166: Function CreateRotationMatrixZ( Sine, Cosine : Single ) : TMatrixGL
11167: Function CreateScaleMatrix( V : TAffineVector ) : TMatrixGL
11168: Function CreateTranslationMatrix( V : TVectorGL ) : TMatrixGL
11169: Procedure MatrixAdjoint( var M : TMatrixGL )
11170: Function MatrixAffineDeterminant( M : TAffineMatrix ) : Single
11171: Procedure MatrixAffineTranspose( var M : TAffineMatrix )
11172: Function MatrixDeterminant( M : TMatrixGL ) : Single
11173: Procedure MatrixInvert( var M : TMatrixGL )
11174: Function MatrixMultiply( M1, M2 : TMatrixGL ) : TMatrixGL
11175: Procedure MatrixScale( var M : TMatrixGL; Factor : Single )
11176: Procedure MatrixTranspose( var M : TMatrixGL )
11177: Function QuaternionConjugate( Q : TQuaternion ) : TQuaternion
11178: Function QuaternionFromPoints( V1, V2 : TAffineVector ) : TQuaternion
11179: Function QuaternionMultiply( qL, qR : TQuaternion ) : TQuaternion
11180: Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11181: Function QuaternionToMatrix( Q : TQuaternion ) : TMatrixGL
11182: Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector )
11183: Function ConvertRotation( Angles : TAffineVector ) : TVectorGL
11184: Function CreateRotationMatrix( Axis : TVector3f; Angle : Single ) : TMatrixGL
11185: //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations ) : Boolean
11186: Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix ) : TAffineVector
11187: Function VectorTransform( V : TVector4f; M : TMatrixGL ) : TVector4f;
11188: Function VectorTransform1( V : TVector3f; M : TMatrixGL ) : TVector3f;
11189: Function MakeAffineDblVector( V : array of Double ) : TAffineDblVector
11190: Function MakeDblVector( V : array of Double ) : THomogeneousDblVector
11191: Function MakeAffineVector( V : array of Single ) : TAffineVector
11192: Function MakeQuaternion( Imag : array of Single; Real : Single ) : TQuaternion
11193: Function MakeVector( V : array of Single ) : TVectorGL
11194: Function PointInPolygonGL( xp, yp : array of Single; x, y : Single ) : Boolean
11195: Function VectorAffineDblToFlt( V : TAffineDblVector ) : TAffineVector
11196: Function VectorDblToFlt( V : THomogeneousDblVector ) : THomogeneousVector
11197: Function VectorAffineFltToDbl( V : TAffineVector ) : TAffineDblVector
11198: Function VectorFltToDbl( V : TVectorGL ) : THomogeneousDblVector
11199: Function ArcCosGL( X : Extended ) : Extended

```

```

11200: Function ArcSinGL( X : Extended ) : Extended
11201: Function ArcTan2GL( Y, X : Extended ) : Extended
11202: Function CoTanGL( X : Extended ) : Extended
11203: Function DegToRadGL( Degrees : Extended ) : Extended
11204: Function RadToDegGL( Radians : Extended ) : Extended
11205: Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended )
11206: Function TanGL( X : Extended ) : Extended
11207: Function Turn( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11208: Function Turnl( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11209: Function Pitch( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11210: Function Pitchl( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11211: Function Roll( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11212: Function Rolll( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11213: end;
11214:
11215:
11216: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11217: begin
11218:   Function RegCreateKey( const RootKey : HKEY; const Key, Value : string ) : Longint
11219:   Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string ) : Boolean
11220:   Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string ) : Boolean
11221:   Function RegReadBool( const RootKey : HKEY; const Key, Name : string ) : Boolean
11222:   Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean ) : Boolean
11223:   Function RegReadInteger( const RootKey : HKEY; const Key, Name : string ) : Integer
11224:   Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer ) : Integer
11225:   Function RegReadString( const RootKey : HKEY; const Key, Name : string ) : string
11226:   Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string ) : string
11227:   Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string ) : Int64
11228:   Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64 ) : Int64
11229:   Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean )
11230:   Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer )
11231:   Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string )
11232:   Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64 )
11233:   Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings ) : Boolean
11234:   Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings ) : Boolean
11235:   Function RegHasSubKeys( const RootKey : HKEY; const Key : string ) : Boolean
11236:   Function RegKeyExists( const RootKey : HKEY; const Key : string ) : Boolean
11237:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11238:             +'RunOnce, ekServiceRun, ekServiceRunOnce )'
11239:             +AddClassN(FindClass('TOBJECT'), 'EJclRegistryError'
11240:   Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string ) : Boolean
11241:   Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string ) : Boolean
11242:   Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
11243: Items:TStrings):Bool;
11244:   Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
11245: SaveTo:TStrings):Bool;
11246:   Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11247: end;
11248: begin
11249:   CLSID_StdComponentCategoriesMgr', 'TGUID '{0002E005-0000-0000-C000-000000000046}
11250:   CATID_SafeForInitializing', 'TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11251:   CATID_SafeForScripting', 'TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11252:   icMAX_CATEGORY_DESC_LEN', 'LongInt'{ 128};
11253:   FindClass('TOBJECT'), 'EInvalidParam
11254:   Function IsDCOMInstalled : Boolean
11255:   Function IsDCOMEnabled : Boolean
11256:   Function GetDCOMVersion : string
11257:   Function GetMDACVersion : string
11258:   Function GetMDACVersion2 : string
11259:   Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
11260: VarArray:OleVariant):HRESULT;
11261:   Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HRESULT;
11262:   Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
11263: VarArray:OleVariant):HRESULT;
11264:   Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HRESULT;
11265:   Function CreateComponentCategory( const CatID : TGUID; const sDescription : string ) : HRESULT
11266:   Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HRESULT
11267:   Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HRESULT
11268:   Function ResetIStreamToStart( Stream : IStream ) : Boolean
11269:   Function SizeOfIStreamContents( Stream : IStream ) : Largeint
11270:   Function StreamToVariantArray( Stream : TStream ) : OleVariant;
11271:   Function StreamToVariantArrayl( Stream : IStream ) : OleVariant;
11272:   Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream );
11273:   Procedure VariantArrayToStreaml( VarArray : OleVariant; var Stream : IStream );
11274:
11275:
11276: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11277: begin
11278:   Const ('CelsiusFreezingPoint','Extended').setExtended( 0.0 );
11279:   FahrenheitFreezingPoint','Extended').setExtended( 32.0 );
11280:   KelvinFreezingPoint','Extended').setExtended( 273.15 );
11281:   CelsiusAbsoluteZero','Extended').setExtended( - 273.15 );
11282:   FahrenheitAbsoluteZero','Extended').setExtended( - 459.67 );
11283:   KelvinAbsoluteZero','Extended').setExtended( 0.0 );

```

```
11284: DegPerCycle', 'Extended').setExtended( 360.0);
11285: DegPerGrad', 'Extended').setExtended( 0.9);
11286: DegPerRad', 'Extended').setExtended( 57.295779513082320876798154814105);
11287: GradPerCycle', 'Extended').setExtended( 400.0);
11288: GradPerDeg', 'Extended').setExtended( 1.111111111111111111111111111111111111111);
11289: GradPerRad', 'Extended').setExtended( 63.661977236758134307553505349006);
11290: RadPerCycle', 'Extended').setExtended( 6.283185307179586476925286766559);
11291: RadPerDeg', 'Extended').setExtended( 0.017453292519943295769236907684886);
11292: RadPerGrad', 'Extended').setExtended( 0.015707963267948966192313216916398);
11293: CyclePerDeg', 'Extended').setExtended( 0.0027777777777777777777777777777777777777777);
11294: CyclePerGrad', 'Extended').setExtended( 0.0025);
11295: CyclePerRad', 'Extended').setExtended( 0.15915494309189533576888376337251);
11296: ArcMinutesPerDeg', 'Extended').setExtended( 60.0);
11297: ArcSecondsPerArcMinute', 'Extended').setExtended( 60.0);
11298: Function HowAOneLinerCanBiteYou( const Step, Max : Longint ) : Longint
11299: Function MakePercentage( const Step, Max : Longint ) : Longint
11300: Function CelsiusToKelvin( const T : double) : double
11301: Function CelsiusToFahrenheit( const T : double) : double
11302: Function KelvinToCelsius( const T : double) : double
11303: Function KelvinToFahrenheit( const T : double) : double
11304: Function FahrenheitToCelsius( const T : double) : double
11305: Function FahrenheitToKelvin( const T : double) : double
11306: Function CycleToDeg( const Cycles : double) : double
11307: Function CycleToGrad( const Cycles : double) : double
11308: Function CycleToRad( const Cycles : double) : double
11309: Function DegToCycle( const Degrees : double) : double
11310: Function DegToGrad( const Degrees : double) : double
11311: Function DegToRad( const Degrees : double) : double
11312: Function GradToCycle( const Grads : double) : double
11313: Function GradToDeg( const Grads : double) : double
11314: Function GradToRad( const Grads : double) : double
11315: Function RadToCycle( const Radians : double) : double
11316: Function RadToDeg( const Radians : double) : double
11317: Function RadToGrad( const Radians : double) : double
11318: Function DmsToDeg( const D, M : Integer; const S : double) : double
11319: Function DmsToRad( const D, M : Integer; const S : double) : double
11320: Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11321: Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11322: Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11323: Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11324: Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11325: Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11326: Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11327: Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11328: Function CmToInch( const Cm : double) : double
11329: Function InchToCm( const Inch : double) : double
11330: Function FeetToMetre( const Feet : double) : double
11331: Function MetreToFeet( const Metre : double) : double
11332: Function YardToMetre( const Yard : double) : double
11333: Function MetreToYard( const Metre : double) : double
11334: Function NmToKm( const Nm : double) : double
11335: Function KmToNm( const Km : double) : double
11336: Function KmToSm( const Km : double) : double
11337: Function SmToKm( const Sm : double) : double
11338: Function LitreToGalUS( const Litre : double) : double
11339: Function GalusToLitre( const GalUs : double) : double
11340: Function GalUsToGalCan( const GalUs : double) : double
11341: Function GalCanToGalUs( const GalCan : double) : double
11342: Function GalUsToGalUK( const GalUs : double) : double
11343: Function GalUKToGalUs( const GalUK : double) : double
11344: Function LitreToGalCan( const Litre : double) : double
11345: Function GalCanToLitre( const GalCan : double) : double
11346: Function LitreToGalUK( const Litre : double) : double
11347: Function GalUKToLitre( const GalUK : double) : double
11348: Function KgToLb( const Kg : double) : double
11349: Function LbToKg( const Lb : double) : double
11350: Function KgToOz( const Kg : double) : double
11351: Function OzToKg( const Oz : double) : double
11352: Function CwtUsToKg( const Cwt : double) : double
11353: Function CwtUKToKg( const Cwt : double) : double
11354: Function KaratToKg( const Karat : double) : double
11355: Function KgToCwtUs( const Kg : double) : double
11356: Function KgToCwtUk( const Kg : double) : double
11357: Function KgToKarat( const Kg : double) : double
11358: Function KgToSton( const Kg : double) : double
11359: Function KgToLton( const Kg : double) : double
11360: Function StonToKg( const STon : double) : double
11361: Function LtonToKg( const Lton : double) : double
11362: Function QrUsToKg( const Qr : double) : double
11363: Function QrUkToKg( const Qr : double) : double
11364: Function KgToQrUs( const Kg : double) : double
11365: Function KgToQrUk( const Kg : double) : double
11366: Function PascalToBar( const Pa : double) : double
11367: Function PascalToAt( const Pa : double) : double
11368: Function PascalToTorr( const Pa : double) : double
11369: Function BarToPascal( const Bar : double) : double
11370: Function AtToPascal( const At : double) : double
11371: Function TorrToPascal( const Torr : double) : double
11372: Function KnotToMs( const Knot : double) : double
```

```

11373: Function HpElectricToWatt( const HpE : double ) : double
11374: Function HpMetricToWatt( const HpM : double ) : double
11375: Function MsToKnot( const ms : double ) : double
11376: Function WattToHpElectric( const W : double ) : double
11377: Function WattToHpMetric( const W : double ) : double
11378: function getBigPI: string; //PI of 1000 numbers
11379:
11380: procedure SIRegister_devutils(CL: TPSPPascalCompiler);
11381: begin
11382:   Function CDEExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ) : THandle
11383:   Procedure CDCopyFile( const FileName, DestName : string )
11384:   Procedure CDMoveFile( const FileName, DestName : string )
11385:   Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor ) : TColor
11386:   Procedure CDDeleteFiles( Sender : TObject; s : string )
11387:   Function CDGetTempDir : string
11388:   Function CDGetFileSize( FileName : string ) : longint
11389:   Function GetfileTime( FileName : string ) : longint
11390:   Function GetShortName( FileName : string ) : string
11391:   Function GetFullName( FileName : string ) : string
11392:   Function WinReboot : boolean
11393:   Function WinDir : string
11394:   Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean ) : cardinal
11395:   Function Runfile_( Cmd, WorkDir : string; Wait : boolean ) : Boolean
11396:   Function devExecutor : TdevExecutor
11397: end;
11398:
11399: procedure SIRegister_FileAssocs(CL: TPSPPascalCompiler);
11400: begin
11401:   Procedure CheckAssociations // AssociationsCount', 'LongInt'( 7 );
11402:   Procedure Associate( Index : integer )
11403:   Procedure UnAssociate( Index : integer )
11404:   Function IsAssociated( Index : integer ) : boolean
11405:   Function CheckFiletype( const extension, filetype, description, verb, serverapp : string ) : boolean
11406:   Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp, IcoNum: string )
11407:   Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string )
11408:   procedure RefreshIcons;
11409:   function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11410:   function MergColor(Colors: Array of TColor): TColor;
11411:   function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11412:   procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11413:   function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11414:   function GetInverseColor(AColor: TColor): TColor;
11415:   procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11416:   procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X,Y: integer;ShadowColor: TColor);
11417:   procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11418:   Procedure GetSystemMenuFont(Font: TFont);
11419: end;
11420:
11421: //*****unit uPSI_JvHLParse;*****
11422: function IsStringConstant(const St: string): Boolean;
11423: function IsIntConstant(const St: string): Boolean;
11424: function IsRealConstant(const St: string): Boolean;
11425: function IsIdentifier(const ID: string): Boolean;
11426: function GetStringValue(const St: string): string;
11427: procedure ParseString(const S: string; Ss: TStrings);
11428: function IsStringConstantW(const St: WideString): Boolean;
11429: function IsIntConstantW(const St: WideString): Boolean;
11430: function IsRealConstantW(const St: WideString): Boolean;
11431: function IsIdentifierW(const ID: WideString): Boolean;
11432: function GetStringValueW(const St: WideString): WideString;
11433: procedure ParseStringW(const S: WideString; Ss: TStrings);
11434:
11435:
11436: //*****unit uPSI_JclMapi;*****
11437:
11438: Function JclSimpleSendMail( const ARecipient, AName, ASubject, ABody : string; const AAttachment : TFileName; ShowDialog : Boolean; AParentWND : HWnd ) : Boolean
11439: Function JclSimpleSendFax( const ARecipient, AName, ASubject, ABody : string; const AAttachment : TFileName; ShowDialog : Boolean; AParentWND : HWnd ) : Boolean
11440: Function JclSimpleBringUpSendMailDialog(const ASubject, ABody:string;const AAttach:TFileName;AParentWND:HWnd):Bool
11441: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean ) : DWORD
11442: Function MapiErrorMessage( const ErrorCode : DWORD ) : string
11443:
11444: procedure SIRegister_IdNTLM(CL: TPSPPascalCompiler);
11445: begin
11446:   //Pdes_key_schedule', '^des_key_schedule // will not work
11447:   Function BuildType1Message( ADomain, AHost : String ) : String
11448:   Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11449:   Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIIdAuthenticationClass)
11450:   Function FindAuthClass( AuthName : String ) : TIIdAuthenticationClass
11451:   GBBase64CodeTable', 'string'ABCDEFIGHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11452:   GXXECodeTable', 'string'`+0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11453:   GUUECodeTable', 'string`^#$%&'()*)+,-./0123456789:;<=>?@ABCDEFGHIJKLM NOPQRSTUVWXYZ[\]^_
11454: end;
11455:
11456: procedure SIRegister_WDosSocketUtils(CL: TPSPPascalCompiler);
11457: begin
11458:   ('IpAny', 'LongWord').SetUIInt( $00000000 );

```

```

11459:  IpLoopBack', 'LongWord').SetUInt( $7F000001);
11460:  IpBroadcast', 'LongWord').SetUInt( $FFFFFFFF);
11461:  IpNone', 'LongWord').SetUInt( $FFFFFFFF);
11462:  PortAny', 'LongWord( $0000);
11463:  SocketMaxConnections', 'LongInt'( 5);
11464:  TIpAddr', 'LongWord
11465:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11466:  Function HostToNetLong( HostLong : LongWord) : LongWord
11467:  Function HostToNetShort( HostShort : Word) : Word
11468:  Function NetToHostLong( NetLong : LongWord) : LongWord
11469:  Function NetToHostShort( NetShort : Word) : Word
11470:  Function StrToIp( Ip : string) : TIpAddr
11471:  Function IpToStr( Ip : TIpAddr) : string
11472: end;
11473:
11474: (*-----*)
11475: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11476: begin
11477:  TAISmtClientAuthType', '( AlsmtClientAuthNone, alsmtClientAut'
11478:  +'hPlain, AlsmtClientAuthLogin, AlsmtClientAuthCramMD5, AlsmtClientAuthCr'
11479:  +'amShal, AlsmtClientAuthAutoSelect )
11480:  TAISmtClientAuthTypeSet', 'set of TAISmtClientAuthType
11481:  SIRegister_TAISmtClient(CL);
11482: end;
11483:
11484: procedure SIRegister_WDOSPlcUtils(CL: TPSPascalCompiler);
11485: begin
11486:  'TBitNo', 'Integer
11487:  TStByteNo', 'Integer
11488:  TStationNo', 'Integer
11489:  TInOutNo', 'Integer
11490:  TIO', '( EE, AA, NE, NA )
11491:  TBitSet', 'set of TBitNo
11492:  TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11493:  TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11494:  TBitAddr', 'LongInt
11495:  TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11496:  TByteAddr', 'SmallInt
11497:  TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11498:  Function BitAddr(aIo: TIO; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11499:  Function BusBitAddr(aIo:TIO;aInoutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11500:  Procedure BitAddrToValues(aBitAddr:TBitAddr;var aIo:TIO;var aInoutNo:TInOutNo;var aByteNo:Byte;var
aBitNo:TBitNo);
11501:  Function BitAddrToStr( Value : TBitAddr) : string
11502:  Function StrToBitAddr( const Value : string) : TBitAddr
11503:  Function ByteAddr( aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11504:  Function BusByteAddr(aIo:TIO;aInoutNo:TInOutNo;aStation:TStatNo;aStByteNo:TStByteNo):TByteAddr
11505:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIO;var aInoutNo:TInOutNo;var aByteNo:Byte)
11506:  Function ByteAddrToStr( Value : TByteAddr) : string
11507:  Function StrToByteAddr( const Value : string) : TByteAddr
11508:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11509:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11510:  Function InOutStateToStr( State : TInOutState) : string
11511:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11512:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11513: end;
11514:
11515: procedure SIRegister_WDOSTimers(CL: TPSPascalCompiler);
11516: begin
11517:  TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048,
11518:  +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11519:  Dpm1PmVector', 'Int64
11520:  'DInterval','LongInt'( 1000);
11521:  // 'Enabled', 'Boolean')BoolToStr( True);
11522:  'DIntFreq', 'string' if64
11523:  // 'Messages', 'Boolean if64';
11524:  SIRegister_TwdxCustomTimer(CL);
11525:  SIRegister_TwdxTimer(CL);
11526:  SIRegister_TwdxRtcTimer(CL);
11527:  SIRegister_TCustonIntTimer(CL);
11528:  SIRegister_TIntTimer(CL);
11529:  SIRegister_TRtcIntTimer(CL);
11530:  Function RealNow : TDateTime
11531:  Function MsToDateTIme( Millisecond : LongInt) : TDateTime
11532:  Function DateTImeToMs( Time : TDateTime) : LongInt
11533: end;
11534:
11535: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11536: begin
11537:  TIIdSyslogPRI', 'Integer
11538:  TIIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11539:  +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11540:  +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11541:  +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11542:  +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11543:  TIIdSyslogSeverity', '(slEmergency, slAlert, slCritical, slError, slWarning, slNotice, slInformational, slDebug)
11544:  SIRegister_TIIdSysLogMsgPart(CL);
11545:  SIRegister_TIIdSysLogMessage(CL);
11546:  Function FacilityToString( AFac : TIIdSyslogFacility) : string

```

```

11547: Function SeverityToString( ASec : TIdSyslogSeverity ) : string
11548: Function NoToSeverity( ASev : Word ) : TIdSyslogSeverity
11549: Function logSeverityToNo( ASev : TIdSyslogSeverity ) : Word
11550: Function NoToFacility( AFac : Word ) : TIdSyslogFacility
11551: Function logFacilityToNo( AFac : TIdSyslogFacility ) : Word
11552: end;
11553:
11554: procedure SIRegister_TextUtils(CL: TPPascalCompiler);
11555: begin
11556:   'UWhitespace', 'String '(?:\s*)
11557:   Function StripSpaces( const AText : string ) : string
11558:   Function CharCount( const AText : string; Ch : Char ) : Integer
11559:   Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11560:   Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11561: end;
11562:
11563:
11564: procedure SIRegister_ExtPascalUtils(CL: TPPascalCompiler);
11565: begin
11566:   ExtPascalVersion', 'String '0.9.8
11567:   AddTypeS('TBrower', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11568:     +'Opera, brKonqueror, brMobileSafari )
11569:   AddTypes('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11570:   AddTypes('TExtProcedure', 'Procedure
11571:   Function DetermineBrowser( const UserAgentStr : string ) : TBrower
11572:   Function ExtExtract( const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool ):bool;
11573:   Function ExtExplode( Delim : char; const S : string; Separator : char ) : TStringList
11574:   Function FirstDelimiter( const Delimiters, S : string; Offset : integer ) : integer
11575:   Function RPosEx( const Substr, Str : string; Offset : integer ) : integer
11576:   Function CountStr( const Substr, Str : string; UntilStr : string ) : integer
11577:   Function StrToJS( const S : string; UseBR : boolean ) : string
11578:   Function CaseOf( const S : string; const Cases : array of string ) : integer
11579:   Function RCaseOf( const S : string; const Cases : array of string ) : integer
11580:   Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer ) : string
11581:   Function SetPaddings( Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool ):string;
11582:   Function SetMargins( Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool ): string;
11583:   Function ExtBefore( const BeforeS, AfterS, S : string ) : boolean
11584:   Function IsUpperCase( S : string ) : boolean
11585:   Function BeautifyJS( const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11586:   Function BeautifyCSS( const AStyle : string ) : string
11587:   Function LengthRegExp( Rex : string; CountAll : Boolean ) : integer
11588:   Function JSDateToDate( JSDate : string ) : TDate
11589: end;
11590:
11591: procedure SIRegister_JclShell(CL: TPPascalCompiler);
11592: begin
11593:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11594:   TSHDeleteOptions', 'set of TSHDeleteOption
11595:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11596:   TSHRenameOptions', 'set of TSHRenameOption
11597:   Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions ) : Boolean
11598:   Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions ) : Boolean
11599:   Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions ) : Boolean
11600:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11601:   TEnumFolderFlags', 'set of TEnumFolderFlag
11602:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11603:     +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11604:     +'IEnumIdList; Folder : IShellFolder; end
11605:   Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11606:   Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11607:   Procedure SHEnumFolderClose( var F : TEnumFolderRec )
11608:   Function SHEnumFolderNext( var F : TEnumFolderRec ) : Boolean
11609:   Function GetSpecialFolderLocation( const Folder : Integer ) : string
11610:   Function DisplayPropDialog( const Handle : HWND; const FileName : string ) : Boolean;
11611:   Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList ) : Boolean;
11612:   Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint ) : Boolean
11613:   Function OpenFolder( const Path : string; Parent : HWND ) : Boolean
11614:   Function OpenSpecialFolder( FolderID : Integer; Parent : HWND ) : Boolean
11615:   Function SHReallocMem( var P : Pointer; Count : Integer ) : Boolean
11616:   Function SHAllocMem( out P : Pointer; Count : Integer ) : Boolean
11617:   Function SHGetMem( var P : Pointer; Count : Integer ) : Boolean
11618:   Function SHFreeMem( var P : Pointer ) : Boolean
11619:   Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder ) : PItemIdList
11620:   Function PathToPidl( const Path : string; Folder : IShellFolder ) : PItemIdList
11621:   Function PathToPidlBind( const FileName : string; out Folder : IShellFolder ) : PItemIdList
11622:   Function PidlBindToParent( const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList ):Bool;
11623:   Function PidlCompare( const Pid1, Pid2 : PItemIdList ) : Boolean
11624:   Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList ) : Boolean
11625:   Function PidlFree( var IdList : PItemIdList ) : Boolean
11626:   Function PidlGetDepth( const Pidl : PItemIdList ) : Integer
11627:   Function PidlGetLength( const Pidl : PItemIdList ) : Integer
11628:   Function PidlGetNext( const Pidl : PItemIdList ) : PItemIdList
11629:   Function PidlToPath( IdList : PItemIdList ) : string
11630:   Function StrRetFreeMem( StrRet : TStrRet ) : Boolean
11631:   Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean ) : string
11632:   PShellLink', '^TShellLink // will not work
11633:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11634:     +'ingDirectory : string; IdList : PItemIdList; Target : string; Description '
11635:     +' : string; IconLocation : string; IconIndex : Integer; HotKey : Word; end

```

```

11636: Procedure ShellLinkFree( var Link : TShellLink)
11637: Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11638: Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11639: Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11640: Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11641: Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11642: Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11643: Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11644: Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11645: Function OverlayIconShared( var Large, Small : HICON) : Boolean
11646: Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11647: Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11648: Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11649: Function ShellExecAndWait(const FileName:string;const Params: string;const Verb:string;CmdShow:Int):Bool;
11650: Function ShellOpenAs( const FileName : string) : Boolean
11651: Function ShellRasDial( const EntryName : string) : Boolean
11652: Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11653: Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11654: TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )'
11655: Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11656: Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11657: Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11658: Function OemKeyScan( wOemChar : Word) : DWORD
11659: Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11660: end;
11661:
11662: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11663: begin
11664:   xmlVersion', 'String '1.0 FindClass('TOBJECT'),'Exml
11665:   //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11666:   Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11667:   Function xmlValidChar2( const Ch : WideChar) : Boolean;
11668:   Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11669:   Function xmlIsLetter( const Ch : WideChar) : Boolean
11670:   Function xmlIsDigit( const Ch : WideChar) : Boolean
11671:   Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11672:   Function xmlIsNameChar( const Ch : WideChar) : Boolean
11673:   Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11674:   Function xmlValidName( const Text : UnicodeString) : Boolean
11675:   //xmlSpace', 'Char #$20 or #$9 or #$D or #$A';
11676:   //Function xmlSkipSpace( var P : PWideChar) : Boolean
11677:   //Function xmlSkipEq( var P : PWideChar) : Boolean
11678:   //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11679:   //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
11680:     : TUnicodeCodeClass
11681:   Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11682:   Function xmlTag( const Tag : UnicodeString) : UnicodeString
11683:   Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11684:   Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11685:   Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11686:   Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11687:   Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11688:   Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11689:   Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11690:   Function xmlComment( const Comment : UnicodeString) : UnicodeString
11691:   Procedure SelfTestcXMLFunctions
11692: end;
11693: (*-----*)
11694: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11695: begin
11696:   Function AWaitCursor : IUnknown
11697:   Function ChangeCursor( NewCursor : TCursor) : IUnknown
11698:   Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11699:   Function YesNo( const ACaption, AMsg : string) : boolean
11700:   Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11701:   Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11702:   Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11703:   Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11704:   Procedure GetSystemPaths( Strings : TStrings)
11705:   Procedure MakeEditNumeric( EditHandle : integer)
11706: end;
11707:
11708: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11709: begin
11710:   AddTypeS('TVideoCodec', '(vcUnknown,vcRGB,vcUYY2,vcUYVY,vcBTYUV,vcYY,U9,vcYUV12,vcY8,vcY211)
11711:   'BI_YUY2','LongWord( $32595559);
11712:   'BI_UYYV','LongWord').SetUInt( $59565955);
11713:   'BI_BTUV','LongWord').SetUInt( $50313459);
11714:   'BI_YVU9','LongWord').SetUInt( $39555659);
11715:   'BI_YUV12','LongWord( $30323449);
11716:   'BI_Y8','LongWord').SetUInt( $20203859);
11717:   'BI_Y211','LongWord').SetUInt( $31313259);
11718:   Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11719:   Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11720: end;
11721:
11722: (*-----*)
11723: procedure SIRegister_AviCap(CL: TPSPascalCompiler);

```

```

11724: begin
11725:   'WM_USER','LongWord').SetUInt( $0400);
11726:   'WM_CAP_START','LongWord').SetUInt($0400);
11727:   'WM_CAP_END','longword').SetUInt($0400+85);
11728: //WM_CAP_START+ 85
11729: // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11730: Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11731: Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11732: Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11733: Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11734: Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11735: Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11736: Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11737: Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11738: Function capGetUserData( hwnd : THandle) : LongInt
11739: Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11740: Function capDriverDisconnect( hwnd : THandle) : LongInt
11741: Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11742: Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11743: Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11744: Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11745: Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11746: Function capfileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11747: Function capfileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11748: Function capfileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11749: Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11750: Function capEditCopy( hwnd : THandle) : LongInt
11751: Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11752: Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11753: Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11754: Function capDlgVideoFormat( hwnd : THandle) : LongInt
11755: Function capDlgVideoSource( hwnd : THandle) : LongInt
11756: Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11757: Function capDlgVideoCompression( hwnd : THandle) : LongInt
11758: Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11759: Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11760: Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11761: Function capPreview( hwnd : THandle; f : Word) : LongInt
11762: Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11763: Function capOverlay( hwnd : THandle; f : Word) : LongInt
11764: Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11765: Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11766: Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11767: Function capGrabFrame( hwnd : THandle) : LongInt
11768: Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11769: Function capCaptureSequence( hwnd : THandle) : LongInt
11770: Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11771: Function capCaptureStop( hwnd : THandle) : LongInt
11772: Function capCaptureAbort( hwnd : THandle) : LongInt
11773: Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11774: Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11775: Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11776: Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11777: Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11778: Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11779: Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11780: Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11781: Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11782: Function capPalettePaste( hwnd : THandle) : LongInt
11783: Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11784: Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11785: //PCapDriverCaps', '^TCapDriverCaps // will not work
11786: TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11787: +' ; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11788: +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11789: +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11790: //PCapStatus', '^TCapStatus // will not work
11791: TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11792: +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11793: +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapfileExists : BO'
11794: +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11795: +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11796: +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11797: +' wNumAudioAllocated : WORD; end
11798: //PCaptureParms', '^TCaptureParms // will not work
11799: TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11800: +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11801: +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11802: +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11803: +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11804: +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11805: +'WMCIStartTime : DWORD; dwMCICount : DWORD; fStepCaptureAt2x : BOOL; wSt'
11806: +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11807: +'he : BOOL; AVStreamMaster : WORD; end
11808: // PCapInfoChunk', '^TCapInfoChunk // will not work
11809: //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11810: 'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11811: 'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11812: Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
  : Integer; hwndParent : THandle; nID : Integer) : THandle

```

```

11813: Function
capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11814: 'IDS_CAP_BEGIN','LongInt'( 300);
11815: 'IDS_CAP_END','LongInt'( 301);
11816: 'IDS_CAP_INFO','LongInt'( 401);
11817: 'IDS_CAP_OUTOFGMEM','LongInt'( 402);
11818: 'IDS_CAP_FILEEXISTS','LongInt'( 403);
11819: 'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11820: 'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11821: 'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11822: 'IDS_CAP_DEFAVIEEXT','LongInt'( 407);
11823: 'IDS_CAP_DEFFPALEXT','LongInt'( 408);
11824: 'IDS_CAP_CANTOPEN','LongInt'( 409);
11825: 'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11826: 'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11827: 'IDS_CAP_VIDEODITERR','LongInt'( 412);
11828: 'IDS_CAP_READONLYFILE','LongInt'( 413);
11829: 'IDS_CAP_WRITEERROR','LongInt'( 414);
11830: 'IDS_CAP_NODISKSPACE','LongInt'( 415);
11831: 'IDS_CAP_SETFILESIZE','LongInt'( 416);
11832: 'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11833: 'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11834: 'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11835: 'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11836: 'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11837: 'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11838: 'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11839: 'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11840: 'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11841: 'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11842: 'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11843: 'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11844: 'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11845: 'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11846: 'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11847: 'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11848: 'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11849: 'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11850: 'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11851: 'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11852: 'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11853: 'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11854: 'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11855: 'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11856: 'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11857: 'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11858: 'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11859: 'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11860: 'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11861: 'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11862: 'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11863: 'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11864: 'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11865: 'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11866: 'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11867: 'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11868: 'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11869: 'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11870: 'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11871: 'AVICAP32','String 'AVICAP32.dll
11872: end;
11873:
11874: procedure SIRegister_ALFcsmMisc(CL: TPSPPascalCompiler);
11875: begin
11876:   Function AlBoolToInt( Value : Boolean) : Integer
11877:   Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11878:   Function AlIsValidEmail( const Value : AnsiString) : boolean
11879:   Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11880:   Function ALInc( var x : integer; Count : integer) : Integer
11881:   function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11882:   function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11883:   procedure ALSaveStringToFile(Str: AnsiString; filename: AnsiString);
11884:   Function ALIsInteger(const S: AnsiString): Boolean;
11885:   function ALIsDecimal(const S: AnsiString): boolean;
11886:   Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11887:   function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11888:   function ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''): AnsiString;
11889:   function ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11890:   function ALUTF8removeBOM(const S: AnsiString): AnsiString;
11891:   Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11892:   Function ALRandomStr(const aLength: Longint): AnsiString;
11893:   Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11894:   Function ALRandomStrU(const aLength: Longint): String;
11895: end;
11896:
11897: procedure SIRegister_ALJSONDoc(CL: TPSPPascalCompiler);
11898: begin
11899:   Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
      aTrueStr: AnsiString; const aFalseStr : AnsiString)

```

```

11900: end;
11901:
11902: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11903: begin
11904:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11905:     +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11906:     +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11907:     +'; ullAvailExtendedVirtual : Int64; end
11908: TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11909: Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX ) : BOOL
11910: Function ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG ) : LONGLONG
11911: 'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ) );
11912: 'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2 );
11913: 'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1 );
11914: 'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8 );
11915: 'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4 );
11916: end;
11917:
11918: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11919: begin
11920:   SIRegister_THandledObject(CL);
11921:   SIRegister_TEvent(CL);
11922:   SIRegister_TMutex(CL);
11923:   SIRegister_TSharedMem(CL);
11924:   'TRACE_BUF_SIZE','LongInt'( 200 * 1024 );
11925:   'TRACE_BUFFER','String 'TRACE_BUFFER
11926:   'TRACE_MUTEX','String 'TRACE_MUTEX
11927:   //PTraceEntry', '^TTraceEntry // will not work
11928:   SIRegister_TIPCTracer(CL);
11929:   'MAX_CLIENTS','LongInt'( 6 );
11930:   'IPCTIMEOUT','LongInt'( 2000 );
11931:   'IPCBUFFER_NAME','String 'BUFFER_NAME
11932:   'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11933:   'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11934:   'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11935:   'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11936:   'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11937:   'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11938:   FindClass('TOBJECT'), 'EMonitorActive
11939:   FindClass('TOBJECT'), 'TIPCThread
11940:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11941:     +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11942:     +'ach, evClientSwitch, evClientSignal, evClientExit )
11943:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11944:   TClientFlags', 'set of TClientFlag
11945:   //PEventData', '^TEventData // will not work
11946:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11947:     +'lag; Flags : TClientFlags; end
11948:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean )
11949:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread )
11950:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData )
11951:   //TIPCEventInfo', '^TIPCEventInfo // will not work
11952:   TIPCEventInfo', 'record FID:Integer; FKind:TEventKind; FData:TEventData; end
11953:   SIRegister_TIPCEvent(CL);
11954:   //PClientDirRecords', '^TClientDirRecords // will not work
11955:   SIRegister_TClientDirectory(CL);
11956:   TIPCState', '( stInactive, stDisconnected, stConnected )
11957:   SIRegister_TIPCThread(CL);
11958:   SIRegister_TIPCMonitor(CL);
11959:   SIRegister_TIPCCClient(CL);
11960:   Function IsMonitorRunning( var Hndl : THandle ) : Boolean
11961:   end;
11962:
11963: (*-----*)
11964: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11965: begin
11966:   SIRegister_TALGSMComm(CL);
11967:   Function ALGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString ) : AnsiString
11968:   Procedure ALGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
11969:   AMessage:AnsiString);
11970:   Function ALGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString ) : AnsiString
11971:   Function ALGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
11972:   UseGreekAlphabet:Bool):Widestring;
11973:   function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11974:   procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11975:   begin
11976:     TALHTTPPropertyChangeEvent', 'Procedure(sender:Tobject;const PropertyInfo:Integer;
11977:     TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11978:     TALHTTPMethod', '( HTTPPmt_Get,HTTPPmt_Post,HTTPPmt_Head,HTTPPmt_Trace,HTTPPmt_Put,HTTPPmt_Delete );
11979:     TIInternetScheme', 'integer
11980:     TALIPv6Binary', 'array[1..16] of Char;
11981:     // TALIPv6Binary = array[1..16] of ansiChar;
11982:     // TIInternetScheme = Integer;
11983:     SIRegister_TALHTTPRequestHeader(CL);
11984:     SIRegister_TALHTTPCookie(CL);
11985:     SIRegister_TALHTTPCookieCollection(CL);
11986:     SIRegister_TALHTTPResponseHeader(CL);

```

```

11987: Function ALHTTPDecode( const AStr : AnsiString ) : AnsiString
11988: Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings )
11989: // Procedure ALExtractHTTPFields( Separators,WhiteSpace, Quotes:TSysCharSet,
Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11990: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11991: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11992: Function AlRemoveSchemeFromUrl( aUrl : AnsiString ) : AnsiString
11993: Function AlExtractSchemeFromUrl( aUrl : AnsiString ) : TIInternetScheme
11994: Function AlExtractHostNameFromUrl( aUrl : AnsiString ) : AnsiString
11995: Function AlExtractDomainNameFromUrl( aUrl : AnsiString ) : AnsiString
11996: Function AlExtractUrlPathFromUrl( aUrl : AnsiString ) : AnsiString
11997: Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
ExtraInfo : AnsiString; var PortNumber : integer ) : Boolean;
11998: Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
Anchor : AnsiString; Query : TALStrings; var PortNumber : integer ) : Boolean;
11999: Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
12000: Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString ) : AnsiString;
12001: Function AlRemoveAnchorFromUrl1( aUrl : AnsiString ) : AnsiString;
12002: Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString ) : AnsiString;
12003: Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
12004: Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
12005: Function ALDateTimeToRfc822Str( const S : AnsiString; out Value : TDateTime ) : Boolean
12006: Function ALRfc822StrToGMTDateTime( const s : AnsiString ) : TDateTime
12007: Function ALRfc822StrToGMTDateTime( const s : AnsiString ) : TDateTime
12008: Function ALTryIPv4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal ) : Boolean
12009: Function ALIPv4StrToNumeric( aIPv4 : ansiString ) : Cardinal
12010: Function ALNumericToIPv4Str( aIPv4 : Cardinal ) : ansiString
12011: Function ALZeroIpV6 : TALIPv6Binary
12012: Function ALTryIPv6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary ) : Boolean
12013: Function ALIPv6StrToBinary( aIPv6 : ansiString ) : TALIPv6Binary
12014: Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary ) : ansiString
12015: Function ALBinaryStrToIPv6Binary( aIPv6BinaryStr : ansiString ) : TALIPv6Binary
12016: end;
12017:
12018: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
12019: begin
12020: Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
DecodeHTMLText:Bool;
12021: Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
12022: Function ALXMLCDataElementEncode( Src : AnsiString ) : AnsiString
12023: Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
12024: Function ALUTF8XMLTextElementDecode( const Src : AnsiString ) : AnsiString
12025: Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHTMLEntities:Bool;const
useNumRef:bool):AnsiString);
12026: Function ALUTF8HTMLDecode( const Src : AnsiString ) : AnsiString
12027: Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
12028: Function ALUTF8JavascriptDecode( const Src : AnsiString ) : AnsiString
12029: Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12030: Procedure ALCompactHtmlTagParams( TagParams : TALStrings )
12031: end;
12032:
12033: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12034: begin
12035: SIRegister_TALEMailHeader(CL);
12036: SIRegister_TALNewsArticleHeader(CL);
12037: Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
decodeRealName:Bool):AnsiString;
12038: Function AlExtractEmailAddress( FriendlyEmail : AnsiString ) : AnsiString
12039: Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString ) : AnsiString
12040: Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString ) : AnsiString
12041: Function ALGenerateInternetMessageID : AnsiString;
12042: Function ALGenerateInternetMessageID1( ahostname : AnsiString ) : AnsiString;
12043: Function ALDecodeQuotedPrintableString( src : AnsiString ) : AnsiString
12044: Function ALDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12045: end;
12046:
12047: (*-----*)
12048: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12049: begin
12050: Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12051: Function ALIPAddrToName( IPAddr : AnsiString ) : AnsiString
12052: Function ALGetLocalIPs : TALStrings
12053: Function ALGetLocalHostName : AnsiString
12054: end;
12055:
12056: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12057: begin
12058: Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
TALWebRequest;ServerVariables:TALStrings);
12059: Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12060: Procedure AlCGIInitDefaultServerVariables( ServerVariables : TALStrings );
12061: Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
ScriptFileName:AnsiString;Url:AnsiStr;
12062: Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
ScriptName, ScriptFileName : AnsiString; Url : AnsiString);

```

```

12063: Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
12064:   : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12065: Procedure AlCGIExec1( ScriptName,ScriptFileName , Url , X_REWRITE_URL , InterpreterFilename:AnsiString;
12066: WebRequest : TALIsapiRequest;
12067: overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'+'overloadedRequestContentStream:Tstream;var
12068: ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12069: Procedure AlCGIExec2( ScriptName,ScriptFileName , Url , X_REWRITE_URL ,
12070: InterpreterFilename:AnsiString;WebRequest : TALIsapiRequest; var ResponseContentString : AnsiString;
12071: ResponseHeader : TALHTTPResponseHeader);
12072: end;
12068:
12069: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12070: begin
12071:   TStartupInfoA', 'TStartupInfo
12072:   'SE_CREATE_TOKEN_NAME', 'String'( 'SeCreateTokenPrivilege
12073: SE_ASSIGNPRIMARYTOKEN_NAME', 'String 'SeAssignPrimaryTokenPrivilege
12074: SE_LOCK_MEMORY_NAME', 'String)( 'SeLockMemoryPrivilege
12075: SE_INCREASE_QUOTA_NAME', 'String 'SeIncreaseQuotaPrivilege
12076: SE_UNSOLICITED_INPUT_NAME', 'String 'SeUnsolicitedInputPrivilege
12077: SE_MACHINE_ACCOUNT_NAME', 'String 'SeMachineAccountPrivilege
12078: SE_TCB_NAME', 'String 'SeTcbPrivilege
12079: SE_SECURITY_NAME', 'String 'SeSecurityPrivilege
12080: SE_TAKE_OWNERSHIP_NAME', 'String 'SeTakeOwnershipPrivilege
12081: SE_LOAD_DRIVER_NAME', 'String 'SeLoadDriverPrivilege
12082: SE_SYSTEM_PROFILE_NAME', 'String 'SeSystemProfilePrivilege
12083: SE_SYSTEMTIME_NAME', 'String 'SeSystemtimePrivilege
12084: SE_PROF_SINGLE_PROCESS_NAME', 'String 'SeProfileSingleProcessPrivilege
12085: SE_INC_BASE_PRIORITY_NAME', 'String 'SeIncreaseBasePriorityPrivilege
12086: SE_CREATE_PAGEFILE_NAME', 'String 'SeCreatePagefilePrivilege
12087: SE_CREATE_PERMANENT_NAME', 'String 'SeCreatePermanentPrivilege
12088: SE_BACKUP_NAME', 'String 'SeBackupPrivilege
12089: SE_RESTORE_NAME', 'String 'SeRestorePrivilege
12090: SE_SHUTDOWN_NAME', 'String 'SeShutdownPrivilege
12091: SE_DEBUG_NAME', 'String 'SeDebugPrivilege
12092: SE_AUDIT_NAME', 'String 'SeAuditPrivilege
12093: SE_SYSTEM_ENVIRONMENT_NAME', 'String 'SeSystemEnvironmentPrivilege
12094: SE_CHANGE_NOTIFY_NAME', 'String 'SeChangeNotifyPrivilege
12095: SE_REMOTE_SHUTDOWN_NAME', 'String 'SeRemoteShutdownPrivilege
12096: SE_UNDOCK_NAME', 'String 'SeUndockPrivilege
12097: SE_SYNC_AGENT_NAME', 'String 'SeSyncAgentPrivilege
12098: SE_ENABLE_DELEGATION_NAME', 'String 'SeEnableDelegationPrivilege
12099: SE_MANAGE_VOLUME_NAME', 'String 'SeManageVolumePrivilege
12100: Function AlGetEnvironmentString : AnsiString
12101: Function ALWinExec32( const FileName,CurrentDir,
12102: Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12103: Function ALWinExec321( const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12104: Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer ) : DWORD
12105: Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer ) : DWORD
12106: Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean ) : Boolean
12107: end;
12108: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12109: begin
12110:   Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
12111: RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12112:   Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
12113: RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime ) : Boolean;
12114:   Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
12115: FileNameMask : ansiString; const FailIfExists : Boolean ) : Boolean
12116:   Function ALGetModuleName : ansistring
12117:   Function ALGetModuleFileNameWithoutExtension : ansistring
12118:   Function ALGetModulePath : ansistring
12119:   Function ALGetFileSize( const AFileName : ansistring ) : int64
12120:   Function ALGetFileVersion( const AFileName : ansistring ) : ansistring
12121:   Function ALGetFileCreationDateTime( const aFileName : Ansistring ) : TDateTime
12122:   Function ALGetFileLastWriteDateTime( const aFileName : Ansistring ) : TDateTime
12123:   Function ALGetFileLastAccessDateTime( const aFileName : Ansistring ) : TDateTime
12124:   Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime )
12125:   Function ALIsDirectoryEmpty( const directory : ansiString ) : boolean
12126:   Function ALFileExists( const Path : ansiString ) : boolean
12127:   Function ALDirectoryExists( const Directory : Ansistring ) : Boolean
12128:   Function ALCreateDir( const Dir : Ansistring ) : Boolean
12129:   Function ALRemoveDir( const Dir : Ansistring ) : Boolean
12130:   Function ALDeleteFile( const FileName : Ansistring ) : Boolean
12131:   Function ALRenameFile( const OldName, NewName : ansistring ) : Boolean
12132: end;
12133: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12134: begin
12135:   NativeInt', 'Integer
12136:   NativeUInt', 'Cardinal
12137:   Function ALMimeBase64EncodeString( const S : AnsiString ) : AnsiString
12138:   Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString ) : AnsiString
12139:   Function ALMimeBase64DecodeString( const S : AnsiString ) : AnsiString
12140:   Function ALMimeBase64EncodedSize( const InputSize : NativeInt ) : NativeInt
12141:   Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt ) : NativeInt
12142:   Function ALMimeBase64DecodedSize( const InputSize : NativeInt ) : NativeInt
12143:   Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
12144: InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )

```

```

12142: Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
12143: InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12144: Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
12145: InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12146: Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
12147: InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12148: + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12149: Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
12150: ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12151: Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
12152: OutputBuf:TByteDynArray);
12153: Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
12154: OutputBuffer:TByteDynArray);
12155: Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
12156: OutputBuffer:TByteDynArray);
12157: Function ALMimeBase64DecodePartial(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
12158: OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12159: Function ALMimeBase64DecodePartialEndl(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
12160: ByteBufferSpace:Cardinal) : NativeInt;
12161: Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12162: Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12163: Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12164: Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12165: Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12166: Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12167: 'cALMimeBase64_ENCODED_LINE_BREAK', 'LongInt'( 76);
12168: 'cALMimeBase64_DECODED_LINE_BREAK', 'LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12169: 'cALMimeBase64_BUFFER_SIZE', 'LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12170: Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12171: Procedure ALFillExtByMimeContentTypeList( AMIMELIST : TALStrings)
12172: Function ALGetDefaultFileExtFromMimeType( aContentType : AnsiString) : AnsiString
12173: Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12174: end;
12175: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12176: begin
12177:   'cALXMLNodeMaxListSize', 'LongInt'( Maxint div 16);
12178:   FindClass('TOBJECT'), 'TALXMLNode
12179:   FindClass('TOBJECT'), 'TALXMLNodeList
12180:   FindClass('TOBJECT'), 'TALXMLDocument
12181:   TALXMLParseProcessingInstructionEvent', 'Procedure (Sender:TObject; const Target,Data:AnsiString)
12182:   TALXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12183:   TALXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12184:   + 'nst Name : AnsiString; const Attributes : TALStrings)
12185:   TALXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12186:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText,
12187:   + 'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument,
12188:   + 'ntDocType, ntDocFragment, ntNotation )
12189:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12190:   TALXMLDocOptions', 'set of TALXMLDocOption
12191:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12192:   TALXMLParseOptions', 'set of TALXMLParseOption
12193:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12194:   PALPointerXMLNodeList', '^PALPointerXMLNodeList // will not work
12195:   SIRegister_EALXMLDocError(CL);
12196:   SIRegister_TALXMLNodeList(CL);
12197:   SIRegister_TALXMLNode(CL);
12198:   SIRegister_TALXmlElementNode(CL);
12199:   SIRegister_TALXmlAttributeNode(CL);
12200:   SIRegister_TALXmlTextNode(CL);
12201:   SIRegister_TALXmlDocumentNode(CL);
12202:   SIRegister_TALXmlCommentNode(CL);
12203:   SIRegister_TALXmlProcessingInstrNode(CL);
12204:   SIRegister_TALXmlCDataNode(CL);
12205:   SIRegister_TALXmlEntityRefNode(CL);
12206:   SIRegister_TALXmlEntityNode(CL);
12207:   SIRegister_TALXmlDocTypeNode(CL);
12208:   SIRegister_TALXmlDocFragmentNode(CL);
12209:   SIRegister_TALXmlNotationNode(CL);
12210:   SIRegister_TALXMLDocument(CL);
12211:   cALMUT8EncodingStr', 'String 'UTF-8
12212:   cALMUT8HeaderStr', 'String <?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>' +#13#10);
12213:   CALNSDelim', 'String :
12214:   CALXML', 'String 'xml
12215:   CALVersion', 'String 'version
12216:   CALEncoding', 'String 'encoding
12217:   CALStandalone', 'String 'standalone
12218:   CALDefaultNodeIndent', 'String '
12219:   CALXmlDocument', 'String 'DOCUMENT
12220:   Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12221:   Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
12222: EncodingStr:AnsiString);
12223:   Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildnodeName,
12224: ChildnodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12225:   Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
12226: ChildnodeName, ChildnodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode

```

```

12218: Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
12219:   Recurse: Boolean):TalxmlNode
12220: Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
12221:   AttributeName,AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12222: Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
12223:   AnsiString
12224: end;
12225:
12226: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12227: //based on TEEProc, TeCanvas, TEEEngine, TChart
12228: begin
12229:   'TeePiStep','Double').setExtended( Pi / 180.0);
12230:   'TeeDefaultPerspective','LongInt'( 100);
12231:   'TeeMinAngle','LongInt'( 270);
12232:   'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12233:   'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12234:   'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12235:   'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0AO ));
12236:   'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12237:   'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12238:   'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12239:   'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0AO ));
12240:   'TA_LEFT','LongInt'( 0);
12241:   'TA_RIGHT','LongInt'( 2);
12242:   'TA_CENTER','LongInt'( 6);
12243:   'TA_TOP','LongInt'( 0);
12244:   'TA_BOTTOM','LongInt'( 8);
12245:   'teePATCOPY','LongInt'( 0);
12246:   'NumCirclePoints','LongInt'( 64);
12247:   'teeDEFAULT_CHARSET','LongInt'( 1);
12248:   'teeANTIALIASED_QUALITY','LongInt'( 4);
12249:   'TA_LEFT','LongInt'( 0);
12250:   'bs_Solid','LongInt'( 0);
12251:   'teePF24Bit','LongInt'( 0);
12252:   'teepfDevice','LongInt'( 1);
12253:   'CM_MOUSELEAVE','LongInt'( 10000);
12254:   'CM_SYSCOLORCHANGE','LongInt'( 10001);
12255:   'DC_BRUSH','LongInt'( 18);
12256:   'DC_PEN','LongInt'( 19);
12257:   'teeCOLORREF','LongWord
12258:   TLogBrush', record lbstyle : Integer; lbColor : TColor; lbHatch: Integer; end
12259: //TNotifyEvent', 'Procedure ( Sender : TObject)
12260: SIRegister_TFilterRegion(CL);
12261: SIRegister_IFormCreator(CL);
12262: SIRegister_TTeeFilter(CL);
12263: //TFilterClass', 'class of TTeeFilter
12264: SIRegister_TFilterItems(CL);
12265: SIRegister_TConvolveFilter(CL);
12266: SIRegister_TBlurFilter(CL);
12267: SIRegister_TTeePicture(CL);
12268: TPenEndStyle', '( esRound, esSquare, esFlat )
12269: SIRegister_TChartPen(CL);
12270: SIRegister_TChartHiddenPen(CL);
12271: SIRegister_TDottedGrayPen(CL);
12272: SIRegister_TDarkGrayPen(CL);
12273: SIRegister_TWhitePen(CL);
12274: SIRegister_TChartBrush(CL);
12275: TTeeView3DSrolled', 'Procedure ( IsHoriz : Boolean)
12276: TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12277: SIRegister_TVview3DOptions(CL);
12278: FindClass('TOBJECT'),TTeeCanvas
12279: TTeeTransparency', 'Integer
12280: SIRegister_TTeeBlend(CL);
12281: FindClass('TOBJECT'),TCanvas3D
12282: SIRegister_TTeeShadow(CL);
12283: TTeeGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
12284: gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12285: FindClass('TOBJECT'),TSubGradient
12286: SIRegister_TCustomTeeGradient(CL);
12287: SIRegister_TSubGradient(CL);
12288: SIRegister_TTeeGradient(CL);
12289: SIRegister_TTeeFontGradient(CL);
12290: SIRegister_TTeeFont(CL);
12291: TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12292: TCanvasTextAlign', 'Integer
12293: TTeeCanvasHandle', 'HDC
12294: SIRegister_TTeeCanvas(CL);
12295: TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12296: SIRegister_TFloatXYZ(CL);
12297: TPoint3D', 'record x : integer; y : integer; z : Integer; end
12298: TRGB', 'record blue: byte; green: byte; red: byte; end
12299: {TRGB=packed record
12300:   Blue : Byte;
12301:   Green : Byte;
12302:   Red : Byte;
12303:   //$$IFDEF CLX //Alpha : Byte; // Linux end;}
12304:   //$$IFDEF CLX //Alpha : Byte; // Linux end;
12305:   //TPoint3D; var Color0, Color1 : TColor) : Boolean

```

```

12303: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12304: TCanvas3DPlane', '( cpX, cpY, cpZ )
12305: //IInterface', 'IUnknown
12306: SIRegister_TCanvas3D(CL);
12307: SIRegister_TTeeCanvas3D(CL);
12308: TTrianglePoints', 'Array[0..2] of TPoint;
12309: TFourPoints', 'Array[0..3] of TPoint;
12310: Function ApplyDark( Color : TColor; HowMuch : Byte ) : TColor
12311: Function ApplyBright( Color : TColor; HowMuch : Byte ) : TColor
12312: Function Point3D( const x, y, z : Integer ) : TPoint3D
12313: Procedure SwapDouble( var a, b : Double )
12314: Procedure SwapInteger( var a, b : Integer )
12315: Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer )
12316: Procedure teeRectCenter( const R : TRect; var X, Y : Integer )
12317: Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer ) : TRect
12318: Function RectFromTriangle( const Points : TTrianglePoints ) : TRect
12319: Function RectangleInRectangle( const Small, Big : TRect ) : Boolean
12320: Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect )
12321: Procedure UnClipCanvas( ACanvas : TCanvas )
12322: Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect )
12323: Procedure ClipRoundRectangle( ACanvas : TTeeCanvas; const Rect : TRect; RoundSize : Integer )
12324: Procedure ClipPolygon( ACanvas : TTeeCanvas; const Points : array of TPoint; NumPoints : Integer )
12325: 'TeeCharForHeight', 'String 'W
12326: 'DarkerColorQuantity', 'Byte').SetUInt( 128 );
12327: 'DarkColorQuantity', 'Byte').SetUInt( 64 );
12328: TButtonGetColorProc', 'Function : TColor
12329: SIRegister_TTeeButton(CL);
12330: SIRegister_TButtonColor(CL);
12331: SIRegister_TComboFlat(CL);
12332: Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TeeCanvasHandle)
12333: Function TeePoint( const aX, aY : Integer ) : TPoint
12334: Function TEEPointInRect( const Rect : TRect; const P : TPoint ) : Boolean;
12335: Function PointInRect1( const Rect : TRect; x, y : Integer ) : Boolean;
12336: Function TeeRect( Left, Top, Right, Bottom : Integer ) : TRect
12337: Function OrientRectangle( const R : TRect ) : TRect
12338: Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer )
12339: Function PolygonBounds( const P : array of TPoint ) : TRect
12340: Function PolygonInPolygon( const A, B : array of TPoint ) : Boolean
12341: Function RGBValue( const Color : TColor ) : TRGB
12342: Function EditColor( AOwner : TComponent; AColor : TColor ) : TColor
12343: Function EditColorDialog( AOwner : TComponent; var AColor : TColor ) : Boolean
12344: Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer ) : TPoint
12345: Function TeeCull( const P : TFourPoints ) : Boolean;
12346: Function TeeCull1( const P0, P1, P2 : TPoint ) : Boolean;
12347: TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12348: Procedure SmoothStretch( Src, Dst : TBitmap );
12349: Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption );
12350: Function TeeDistance( const x, y : Double ) : Double
12351: Function TeeLoadLibrary( const FileName : String ) : HInst
12352: Procedure TeeFreeLibrary( hLibModule : HMODULE )
12353: Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint )
12354: //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap )
12355: Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12356: SIRegister_ICanvasHyperlinks(CL);
12357: SIRegister_ICanvasToolTips(CL);
12358: Function Supports( const Instance : IInterface; const IID : TGUID ) : Boolean
12359: end;
12360:
12361: procedure SIRegister_ovcmisc(CL: TPPascalCompiler);
12362: begin
12363: TOvcHdc', 'Integer
12364: TOvcHWND', 'Cardinal
12365: TOvcHdc', 'HDC
12366: TOvcHWND', 'HWND
12367: Function LoadBaseBitmap( lpBitmapName : PChar ) : HBITMAP
12368: Function LoadBaseCursor( lpCursorName : PChar ) : HCURSOR
12369: Function ovCompStruct( const S1, S2, Size : Cardinal ) : Integer
12370: Function DefaultEpoch : Integer
12371: Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12372: Procedure FixRealPrim( P : PChar; DC : Char )
12373: Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer ) : string
12374: Function GetLeftButton : Byte
12375: Function GetNextDlgItem( Ctrl : TOvcHWnd ) : hWnd
12376: Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte )
12377: Function GetShiftFlags : Byte
12378: Function ovCreateRotatedFont( F : TFont; Angle : Integer ) : hFont
12379: Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12380: Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet ) : string
12381: Function ovIsForegroundTask : Boolean
12382: Function ovTrimLeft( const S : string ) : string
12383: Function ovTrimRight( const S : string ) : string
12384: Function ovQuotedStr( const S : string ) : string
12385: Function ovWordCount( const S : string; const WordDelims : TCharSet ) : Integer
12386: Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12387: Function PtrDiff( const P1, P2 : PChar ) : Word
12388: Procedure PtrInc( var P, Delta : Word )
12389: Procedure PtrDec( var P, Delta : Word )
12390: Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer )

```

```

12391: Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
12392:   SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12393: Function ovMinI( X, Y : Integer ) : Integer
12394: Function ovMaxI( X, Y : Integer ) : Integer
12395: Function ovMinL( X, Y : LongInt ) : LongInt
12396: Function ovMaxL( X, Y : LongInt ) : LongInt
12397: Function GenerateComponentName( PF : TWInControl; const Root : string ) : string
12398: Function PartialCompare( const S1, S2 : string ) : Boolean
12399: Function PathEllipsis( const S : string; MaxWidth : Integer ) : string
12400: Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor ) : TBitmap
12401: Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas )
12401: Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
12401:   TransparentColor : TColor )
12402: Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
12402:   TransparentColor : TColorRef)
12403: Function ovWidthOf( const R : TRect ) : Integer
12404: Function ovHeightOf( const R : TRect ) : Integer
12405: Procedure ovDebugOutput( const S : string )
12406: Function GetArrowWidth( Width, Height : Integer ) : Integer
12407: Procedure StripCharSeq( CharSeq : string; var Str : string )
12408: Procedure StripCharFromEnd( aChr : Char; var Str : string )
12409: Procedure StripCharFromFront( aChr : Char; var Str : string )
12410: Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12411: Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL
12412: Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12413: Function CreateEllipticRgn( p1, p2, p3, p4 : Integer ) : HRGN
12414: Function CreateEllipticRgnIndirect( const p1 : TRect ) : HRGN
12415: Function CreateFontIndirect( const p1 : TLogFont ) : HFONT
12416: Function CreateMetaFile( p1 : PChar ) : HDC
12417: Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12418: Function DrawText(HDC: HDC;lpString: PChar;nCount: Integer; var lpRect : TRect; uFormat:UINT): Integer
12419: Function DrawTextS(hDC: HDC;lpString: string;nCount: Integer; var lpRect: TRect; uFormat:UINT): Integer
12420: Function SetMapperFlags( DC : HDC; Flag : DWORD ) : DWORD
12421: Function SetGraphicsMode( hdc : HDC; iMode : Integer ) : Integer
12422: Function SetMapMode( DC : HDC; p2 : Integer ) : Integer
12423: Function SetMetaFileBitsEx( Size : UINT; const Data : PChar ) : HMETAFILE
12424: //Function SetPaletteEntries(Palette:HPALETTE;startIndex,NumEntries:UINT;var PaletteEntries):UINT
12425: Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF ) : COLORREF
12426: Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF ) : BOOL
12427: //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor ) : BOOL
12428: Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer ) : Integer
12429: Function StretchBlt(DestDC: HDC; X, Y, Width, Height: Int; SrcDC: HDC; XSrc, YSrc, SrcWidth,
12429:   SrcHeight: Int; Rop: WORD): BOOL
12430: Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer ) : BOOL
12431: Function StretchDIBits(DC : HDC; DestX, DestY, DestWidth, DestHeight, SrcX, SrcY, SrcWidth,
12431:   SrcHeight: Int; Bits: int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : WORD) : Integer
12432: Function SetROP2( DC : HDC; p2 : Integer ) : Integer
12433: Function SetStretchBltMode( DC : HDC; StretchMode : Integer ) : Integer
12434: Function SetSystemPaletteUse( DC : HDC; p2 : UINT ) : UINT
12435: Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer ) : Integer
12436: Function SetTextColor( DC : HDC; Color : COLORREF ) : COLORREF
12437: Function SetTextAlign( DC : HDC; Flags : UINT ) : UINT
12438: Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer ) : Integer
12439: Function UpdateColors( DC : HDC ) : BOOL
12440: Function GetViewportExtEx( DC : HDC; var Size : TSize ) : BOOL
12441: Function GetViewportOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12442: Function GetWindowExtEx( DC : HDC; var Size : TSize ) : BOOL
12443: Function GetWindowOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12444: Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer ) : Integer
12445: Function InvertRgn( DC : HDC; p2 : HRGN ) : BOOL
12446: Function MaskBlt(DestDC: HDC; XDest, YDest, Width, Height: Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
12446:   HBITMAP; xMask, yMask : Integer; Rop : WORD) : BOOL
12447: Function PlgBlt(DestDC: HDC; const PtsArray, SrcDC: HDC; XSrc, YSrc, Widt, Heigh: Int; Mask: HBITMAP; xMask,
12447:   yMask: Int): BOOL;
12448: Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer ) : Integer
12449: Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer ) : Integer
12450: Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : WORD ) : BOOL
12451: Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer ) : BOOL
12452: Function PlayMetaFile( DC : HDC; MF : HMETAFILE ) : BOOL
12453: Function PaintRgn( DC : HDC; RGN : HRGN ) : BOOL
12454: Function PtInRegion( RGN : HRGN; X, Y : Integer ) : BOOL
12455: Function PtVisible( DC : HDC; X, Y : Integer ) : BOOL
12456: Function RectInRegion( RGN : HRGN; const Rect : TRect ) : BOOL
12457: Function RectVisible( DC : HDC; const Rect : TRect ) : BOOL
12458: Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer ) : BOOL
12459: Function RestoreDC( DC : HDC; SavedDC : Integer ) : BOOL
12460: end;
12461:
12462: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12463: begin
12464:   SIRegister_TOvcAbstractStore(CL);
12465:   //PEXPropInfo', '^TExPropInfo // will not work
12466:   // _TEXPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12467:   SIRegister_TOvcPropertyList(CL);
12468:   SIRegister_TOvcDataFiler(CL);
12469: Procedure UpdateStoredList( AForm : TWInControl; AStoredList : TStrings; FromForm : Boolean )
12470: Procedure UpdateStoredList( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean )
12471: Function CreateStoredItem( const CompName, PropName : string ) : string
12472: Function ParseStoredItem( const Item : string; var CompName, PropName : string ) : Boolean

```

```

12473: //Function GetPropType( PropInfo : PExPropInfo ) : PTypeInfo
12474: end;
12475:
12476: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12477: begin
12478:   'ovsetsize','LongInt'( 16 );
12479:   'etSyntax','LongInt'( 0 );
12480:   'etSemantic','LongInt'( 1 );
12481:   'chCR','Char #13';
12482:   'chLF','Char #10';
12483:   'chLineSeparator',' chCR');
12484:   SIRegister_TCocoError(CL);
12485:   SIRegister_TCommentItem(CL);
12486:   SIRegister_TCommentList(CL);
12487:   SIRegister_TSymbolPosition(CL);
12488:   TGenListType', '( glNever, glAlways, glOnError )
12489:   TovBitSet', 'set of Integer
12490:   //PStartTable', '^TStartTable // will not work
12491:   'TovCharSet', 'set of AnsiChar
12492:   TAFTERGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12493:   TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12494:   TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12495:   TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12496:   TovErrorProc', 'Procedure ( ErrorCode: integer; Symbol : TSymbolD'
12497:     +'osition; const Data : string; ErrorType : integer)
12498:   TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12499:   TGetCH', 'Function ( pos : longint ) : char
12500:   TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12501:   SIRegister_TCocoRScanner(CL);
12502:   SIRegister_TCocoRGrammar(CL);
12503:   '_EF','Char #0);
12504:   '_TAB','Char').SetString( #09);
12505:   '_CR','Char').SetString( #13);
12506:   '_LF','Char').SetString( #10);
12507:   '_EL','').SetString( _CR);
12508:   '_EOF','Char').SetString( #26);
12509:   'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12510:   'minErrDist','LongInt'( 2 );
12511:   Function ovPadL( S : string; ch : char; L : integer ) : string
12512: end;
12513:
12514: TFormSettings = record
12515:   CurrencyFormat: Byte;
12516:   NegCurrFormat: Byte;
12517:   ThousandSeparator: Char;
12518:   DecimalSeparator: Char;
12519:   CurrencyDecimals: Byte;
12520:   DateSeparator: Char;
12521:   TimeSeparator: Char;
12522:   ListSeparator: Char;
12523:   CurrencyString: string;
12524:   ShortDateFormat: string;
12525:   LongDateFormat: string;
12526:   TimeAMString: string;
12527:   TimePMString: string;
12528:   ShortTimeFormat: string;
12529:   LongTimeFormat: string;
12530:   ShortMonthNames: array[1..12] of string;
12531:   LongMonthNames: array[1..12] of string;
12532:   ShortDayNames: array[1..7] of string;
12533:   LongDayNames: array[1..7] of string;
12534:   TwoDigitYearCenturyWindow: Word;
12535: end;
12536:
12537: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12538: begin
12539:   Function ovFormatSettings : TFormSettings
12540: end;
12541:
12542: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12543: begin
12544:   TOvcCharSet', 'set of Char
12545:   ovTable', 'array[0..255] of Byte
12546:   //BTABLE = array[0..{$IFDEF UNICODE}{$ENDIF}{$ELSE}{$ENDIF}]{${$ENDIF}} of Byte;
HUGE_UNICODE_BMTABLE{$FFFF}{$ELSE}{$ENDIF}{$ELSE}{$ENDIF} of Byte;
12547:   Function BinaryBPChar( Dest : PChar; B : Byte ) : PChar
12548:   Function BinaryLPChar( Dest : PChar; L : LongInt ) : PChar
12549:   Function BinaryWPChar( Dest : PChar; W : Word ) : PChar
12550:   Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable )
12551:   Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12552:   Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12553:   Function CharPChar( Dest : PChar; C : Char; Len : Cardinal ) : PChar
12554:   Function DatabPChar( Dest : PChar; Src : PChar; TabSize : Byte ) : PChar
12555:   Function HexBPChar( Dest : PChar; B : Byte ) : PChar
12556:   Function HexLCChar( Dest : PChar; L : LongInt ) : PChar
12557:   Function HexPtrPChar( Dest : PChar; P : TObject ) : PChar
12558:   Function HexWPChar( Dest : PChar; W : Word ) : PChar
12559:   Function LoCaseChar( C : Char ) : Char
12560:   Function OctalLPChar( Dest : PChar; L : LongInt ) : PChar

```

```

12561: Function StrChDeletePrim( P : PChar; Pos : Cardinal ) : PChar
12562: Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal ) : PChar
12563: Function StrChPos( P : PChar; C : Char; var Pos : Cardinal ) : Boolean
12564: Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12565: Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal ) : PChar
12566: Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal ) : PChar
12567: Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal ) : PChar
12568: Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal ) : PChar
12569: Function StrStPos( P, S : PChar; var Pos : Cardinal ) : Boolean
12570: Function StrToInt64Char( S : PChar; var I : LongInt ) : Boolean
12571: Procedure TrimAllSpacesPChar( P : PChar)
12572: Function TrimEmbeddedZeros( const S : string ) : string
12573: Procedure TrimEmbeddedZerosPChar( P : PChar)
12574: Function TrimTrailPrimPChar( S : PChar ) : PChar
12575: Function TrimTrailingZeros( const S : string ) : string
12576: Procedure TrimTrailingZerosPChar( P : PChar)
12577: Function UpCaseChar( C : Char ) : Char
12578: Function ovcCharInSet( C : Char; const CharSet : TOvc CharSet ) : Boolean
12579: Function ovc32StringIsCurrentCodePage( const S : WideString ) : Boolean;
12580: //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal ) : Boolean;
12581: end;
12583:
12584: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12585: begin
12586:   //PraiseFrame', '^TRaiseFrame // will not work
12587:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : __Poin'
12588:   +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12589:   Procedure SafeCloseHandle( var Handle : THandle )
12590:   Procedure ExchangeInteger( X1, X2 : Integer )
12591:   Procedure FillInteger( const Buffer, Size, Value : Integer )
12592:   Function LongMulDiv( Multi, Mult2, Div1 : Longint ) : Longint
12593:   Function afCompareMem( P1, P2 : TObject; Length : Integer ) : Boolean
12594:
12595:   FILENAME_ADVAPI32 = 'ADVAPI32.DLL';
12596:   function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12597:   function AbortSystemShutdown(lpMachineName: PKOLOChar): BOOL; stdcall;
12598:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLOChar;
12599:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12600:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12601:       const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12602:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12603:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLOChar;
12604:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12605:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12606:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12607:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12608:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12609:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLOChar;
12610:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12611:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12612:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12613:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12614:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12615:   function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLOChar): BOOL; stdcall;
12616:   function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLOChar): BOOL; stdcall;
12617:   function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLOChar;
12618:     lpCommandLine: PKOLOChar; lpProcessAttributes: PSecurityAttributes;
12619:     lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12620:     dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLOChar;
12621:     const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12622:   function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12623:   function GetFileSecurity(lpFileName: PKOLOChar; RequestedInformation: SECURITY_INFORMATION;
12624:     pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12625:   function GetUserName(lpBuffer: PKOLOChar; var nSize: DWORD): BOOL; stdcall;
12626:   function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLOChar;
12627:     dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12628:   function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLOChar;
12629:     dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12630:   function LookupAccountName(lpSystemName, lpAccountName: PKOLOChar;
12631:     Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLOChar;
12632:     var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12633:   function LookupAccountSid(lpSystemName: PKOLOChar; Sid: PSID;
12634:     Name: PKOLOChar; var cbName: DWORD; ReferencedDomainName: PKOLOChar;
12635:     var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12636:   function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLOChar;
12637:     lpDisplayName: PKOLOChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12638:   function LookupPrivilegeName(lpSystemName: PKOLOChar;
12639:     var lpLuid: TLargeInteger; lpName: PKOLOChar; var cbName: DWORD): BOOL; stdcall;
12640:   function LookupPrivilegeValue(lpSystemName, lpName: PKOLOChar;
12641:     var lpLuid: TLargeInteger): BOOL; stdcall;
12642:   function ObjectCloseAuditAlarm(SubsystemName: PKOLOChar;
12643:     HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12644:   function ObjectDeleteAuditAlarm(SubsystemName: PKOLOChar;
12645:     HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12646:   function ObjectOpenAuditAlarm(SubsystemName: PKOLOChar; HandleId: Pointer;
12647:     ObjectTypeName: PKOLOChar; ObjectName: PKOLOChar; pSecurityDescriptor: PSecurityDescriptor;
12648:     ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12649:     var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;

```

```

12650:     var GenerateOnClose: BOOL; stdcall;
12651:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12652:                                         HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12653:                                         var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12654:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12655:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12656:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12657:                                         ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12658:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12659:                            lpBuffer: Pointer; nNumberOfBytesToRead: DWORD);
12660:     var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD; BOOL; stdcall;
12661:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12662:                                var phkResult: HKEY); Longint; stdcall;
12663:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12664:                           var phkResult: HKEY); Longint; stdcall;
12665:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12666:                             Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12667:                             lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12668:                             lpdwDisposition: PDWORD); Longint; stdcall;
12669:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar); Longint; stdcall;
12670:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar); Longint; stdcall;
12671:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12672:                           var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12673:                           lpcbClass: PDWORD; lpftLastWriteTime: PFileTime); Longint; stdcall;
12674:     function RegEnumKey(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar; cbName: DWORD); Longint; stdcall;
12675:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12676:                           var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12677:                           lpData: PByte; lpcbData: PDWORD); Longint; stdcall;
12678:     function RegLoadKey(hKey: HKEY; lpSubKey: PKOLChar); Longint; stdcall;
12679:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar; var phkResult: HKEY); Longint; stdcall;
12680:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12681:                           ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY); Longint; stdcall;
12682:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLchar;
12683:                               lpcbClass: PDWORD; lpReserved: Pointer;
12684:                               lpcbSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12685:                               lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12686:                               lpftLastWriteTime: PFileTime); Longint; stdcall;
12687:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12688:                                     NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD); Longint; stdcall;
12689:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12690:                             lpValue: PKOLChar; var lpcbValue: Longint); Longint; stdcall;
12691:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12692:                               lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD); Longint; stdcall;
12693:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12694:                             lpNewFile: PKOLChar; lpOldFile: PKOLChar); Longint; stdcall;
12695:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD); Longint; stdcall;
12696:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12697:                           lpSecurityAttributes: PSecurityAttributes); Longint; stdcall;
12698:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12699:                           dwType: DWORD; lpData: PKOLChar; cbData: DWORD); Longint; stdcall;
12700:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12701:                             Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD); Longint; stdcall;
12702:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar); Longint; stdcall;
12703:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12704:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12705:                           dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12706:                           dwDataSize: DWORD; lpStrings, lpRawData: Pointer); BOOL; stdcall;
12707:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12708:                               pSecurityDescriptor: PSecurityDescriptor); BOOL; stdcall;
12709:
12710:     Function wAddAtom( lpString : PKOLchar ) : ATOM
12711:     Function wBeginUpdateResource( pFileName : PKOLchar; bDeleteExistingResources : BOOL ) : THandle
12712:     //Function wCallNamedPipe( lpNamedPipeName : PKOLchar; lpInBuffer : Pointer; nInBufferSize : DWORD;
12713:     lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD ) : BOOL
12714:     //Function wCommConfigDialog( lpszName : PKOLchar; hWnd : HWND; var lpCC : TCommConfig ) : BOOL
12715:     Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLchar; cchCount1 : Integer;
12716:                               lpString2 : PKOLchar; cchCount2 : Integer ) : Integer
12717:     Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLchar; bFailIfExists : BOOL ) : BOOL
12718:     //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLchar; lpProgressRoutine :
12719:     TFnProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD ) : BOOL
12720:     Function wCreateDirectory( lpPathName : PKOLchar; lpSecurityAttributes : PSecurityAttributes ) : BOOL
12721:     Function wCreateDirectoryEx(lpTemplateDirectory,
12722:                               lpNewDirectory:PKOLchar;lpSecAttrib:PSecurityAttribs):BOOL
12723:     Function wCreateEvent(lpEventAttribs:PSecurityAttrib/bManualReset,
12724:                           bInitialState:BOOL;lpName:PKOLchar):THandle
12725:     Function wCreateFile( lpFileName : PKOLchar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes : PSecurityAttributes;
12726:                           dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12727:     Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
12728:                               dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLchar ) : THandle
12729:     Function wCreateHardLink(lpFileName,
12730:                            lpExistingFileName:PKOLchar;lpSecurityAttributes:PSecurityAttributes):BOOL
12731:     Function CreateMailslot(lpName:PKOLchar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle;
12732:     Function wCreateNamedPipe( lpName : PKOLchar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
12733:                               nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes ) : THandle
12734:     //Function CreateProcess( lpApplicationName : PKOLchar; lpCommandLine : PKOLchar; lpProcessAttributes,
12735:                             lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
12736:                             Pointer;lpCurrentDirectory:PKOLchar;const lpStartupInfo:TStartupInfo;var
12737:                             lpProcessInfo:TProcessInformation):BOOL

```

```

12726: Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
12727:     Longint; lpName : PKOLChar) : THandle
12728: Function wCreateWaitableTimer(lpTimerAttribs: PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar) : THandle;
12729: Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12730: Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12731: //Function
12732: //Function wEnumDateFormats(lpDateFormatEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12733: //Function
12734: //Function wEnumResourceNames(hModule: HMODULE; lpType: PKOLChar; lpEnumFunc: ENUMRESNAMEPROC; lParam: Longint) : BOOL;
12735: //Function wEnumResourceTypes( hModule: HMODULE; lpEnumFunc: ENUMRESTYPEPROC; lParam: Longint) : BOOL,
12736: //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12737: //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12738: //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc; Locale:LCID;dwFlags:DWORD) : BOOL;
12739: Procedure wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12740: //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
12741: dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12742: Function wFindAtom( lpString : PKOLChar) : ATOM
12743: Function wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD) : THandle;
12744: Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12745: //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFIndexInfoLevels; lpFindFileData :
12746: Pointer; fSearchOp : TFIndexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12747: Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12748: Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12749: Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12750: Function wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer) : Integer;
12751: //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
12752: DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12753: Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12754: Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12755: Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12756: Function wGetCommandLine : PKOLChar
12757: //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12758: Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12759: Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12760: //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
12761: PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12762: Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12763: //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12764: Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
12765: lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12766: //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
12767: lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12768: Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12769: Function wGetEnvironmentStrings : PKOLChar
12770: Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12771: Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12772: //Function
12773: wGetFileAttributesEx( lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInfor:Pointer):BOOL;
12774: Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
12775: lpFilePart:PKOLChar):DWORD;
12776: //Function wGetLocaleInfo(Locale:LCID; LCTYPE:LCTYPE;lpLCDData:PKOLChar;cchData:Integer): Integer
12777: Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12778: Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12779: Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12780: //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
12781: lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12782: //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
12783: lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12784: Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12785: Function wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12786: Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12787: Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
12788: nSize:DWORD; lpFileName : PKOLChar) : DWORD
12789: Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12790: Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12791: Function wGetProfileString(lpAppName,lpKeyName,
12792: lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12793: Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12794: //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12795: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
12796: lpCharType):BOOL
12797: Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12798: Function wGetTempFileName( lpPathName, lpPrefixString : PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12799: Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12800: //Function
12801: wGetTimeFormat(Loc:LCID,dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12802: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12803: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
12804: : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
12805: lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL

```

```

12791: Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT ) : UINT
12792: Function wGlobalAddAtom( lpString : PKOLChar ) : ATOM
12793: Function wGlobalFindAtom( lpString : PKOLChar ) : ATOM
12794: Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer ) : UINT
12795: Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UInt ) : BOOL
12796: Function
wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12797: Function wLoadLibrary( lpLibFileName : PKOLChar ) : HMODULE
12798: Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD ) : HMODULE
12799: Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar ) : BOOL
12800: Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD ) : BOOL
12801: //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
TfnProgressRoutine; lpData : Pointer; dwFlags : DWORD ) : BOOL
12802: Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar ) : THandle
12803: Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar ) : THandle
12804: Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar ) : THandle
12805: Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar ) : THandle
12806: Function wOpenWaitableTimer( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpTimerName : PKOLChar ) : THandle
12807: Procedure wOutputDebugString( lpOutputString : PKOLChar )
12808: //Function wPeekConsoleInput( hConsoleInput : THandle; var lpBuffer : TInputRecord; nLength : DWORD; var
lpNumberOfEventsRead : DWORD ) : BOOL;
12809: Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD ) : DWORD
12810: //Function wQueryRecoveryAgents( p1 : PKOLChar; var p2 : Pointer; var p3 : TRecoveryAgentInformation ) : DWORD
12811: //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
lpNumberOfCharsRead : DWORD; lpReserved : Pointer ) : BOOL
12812: //Function wReadConsoleInput( hConsInp : THandle; var lpBuf : TInpRec; nLength : DWORD; var
lpNumbOfEventsRead : DWORD ) : BOOL;
12813: //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
: TCoord; var lpReadRegion : TSmallRect ) : BOOL
12814: //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD ) : BOOL
12815: Function wRemoveDirectory( lpPathName : PKOLChar ) : BOOL
12816: //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo ) : BOOL
12817: Function wSearchPath( lpPath, lpFileName, lpExtension : PKOLChar; nBufferLength : DWORD; lpBuffer : PKOLChar; var
lpFilePart : PKOLChar ) : DWORD;
12818: Function wSetComputerName( lpComputerName : PKOLChar ) : BOOL
12819: Function wSetConsoleTitle( lpConsoleTitle : PKOLChar ) : BOOL
12820: Function wSetCurrentDirectory( lpPathName : PKOLChar ) : BOOL
12821: //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCCommConfig; dwSize : DWORD ) : BOOL
12822: Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar ) : BOOL
12823: Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD ) : BOOL
12824: //Function wSetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCDData : PKOLChar ) : BOOL
12825: Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar ) : BOOL
12826: //Function wUpdateResource( hUpdate : THandle; lpType,
lpName : PKOLChar; wLanguage : Word; lpData : Ptr; cbData : DWORD ) : BOOL
12827: Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD ) : DWORD
12828: Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD ) : BOOL
12829: //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer ) : BOOL
12830: //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
var lpNumberOfEventsWritten : DWORD ) : BOOL
12831: //Function wWriteConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord :
TCoord; var lpWriteRegion : TSmallRect ) : BOOL
12832: //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12833: Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar ) : BOOL
12834: Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar ) : BOOL
12835: Function wWriteProfileSection( lpAppName, lpString : PKOLChar ) : BOOL
12836: Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar ) : BOOL
12837: Function wlstrcat( lpString1, lpString2 : PKOLChar ) : PKOLChar
12838: Function wlstrcmp( lpString1, lpString2 : PKOLChar ) : Integer
12839: Function wlstrcmpi( lpString1, lpString2 : PKOLChar ) : Integer
12840: Function wlstrcpy( lpString1, lpString2 : PKOLChar ) : PKOLChar
12841: Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer ) : PKOLChar
12842: Function wlstrlen( lpString : PKOLChar ) : Integer
12843: //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruct :
PNetConnectInfoStruct ) : DWORD
12844: //Function wWNetAddConnection2( var lpNetResource : TNetResource; lpPassw,
lpUserName : PKOLChar; dwFlags : DWORD ) : DWORD;
12845: //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource : TNetResource; lpPassword,
lpUserName : PKOLChar; dwFlags : DWORD ) : DWORD
12846: Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar ) : DWORD
12847: Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL ) : DWORD
12848: Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL ) : DWORD
12849: //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct ) : DWORD
12850: //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct ) : DWORD
12851: //Function wWNetEnumResource( hEnum : THandle; var lpcCount : DWORD; lpBuffer : Ptr; var lpBufferSize : DWORD ) : DWORD;
12852: Function wWNetGetConnection( lpLocalName : PKOLChar; lpRemoteName : PKOLChar; var lpnLength : DWORD ) : DWORD;
12853: Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
: PKOLChar; nNameBufSize : DWORD ) : DWORD
12854: //Function wWNetGetNetworkInformation( lpProvider : PKOLChar; var lpNetInfoStruct : TNetInfoStruct ) : DWORD;
12855: Function wWNetGetProviderName( dwNetType : DWORD; lpProviderName : PKOLChar; var lpBufferSize : DWORD ) : DWORD;
12856: //Function wWNetGetResourceParent( lpNetResource : PNetResource; lpBuffer : Pointer; var cbBuffer : DWORD ) : DWORD;
12857: //Function wWNetGetUniversalName( lpLocalPath : PKOLChar; dwInfoLevel : DWORD; lpBuffer : Ptr; var
lpBufferSize : DWORD );
12858: Function wWNet GetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD ) : DWORD
12859: // Function wWNetOpenEnum( dwScope, dwType, dwUsage : DWORD; lpNetResource : PNetRes; var lphEnum : THandle ) : DWORD;
12860: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer ) : DWORD

```

```

12861: //Function wNetUseConnection(hwndOwner:HWND;var lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar;dwFlags:DWORD;lpAccessName:PKOLChar;var lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12862: Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12863: Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD ) : DWORD
12864: Function wVerFindfile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT ) : DWORD
12865: Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir, szTmpFile : PKOLChar; var lpuTmpFileLen : UINT ) : DWORD
12866: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12867: //Func wGetPrivateProfileStruct(lpszSection,
12868: //Func wWritePrivateProfileStruct(lpszSection,
12869: //Func wAddFontResource(FileName : PKOLChar) : Integer
12870: //Function wAddFontResourceEx( p1 : PKOLChar; p2 : WORD; p3 : PDesignVector) : Integer
12871: Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar ) : HENHMETAFILE
12872: Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar ) : HMETAFILE
12873: //Function wCreateColorSpace( var ColorSpace : TLogColorSpace ) : HCOLORSPACE
12874: //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode ) : HDC
12875: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar ) : HDC
12876: Function wCreateFont( nHeight, nWidth, nEscapement, nOrientation, fnWeight : Integer; fdwItalic, fdwUnderline, fdwStrikeOut, fdwCharSet, fdwOutputPrec, fdwClipPrecision, fdwQualy, fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT
12877: Function wCreateFontIndirect( const p1 : TLogFont ) : HFONT
12878: //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV ) : HFONT
12879: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode ) : HDC
12880: Function wCreateMetaFile( p1 : PKOLChar ) : HDC
12881: Function wCreateScalableFontResource( p1 : WORD; p2, p3, p4 : PKOLChar ) : BOOL
12882: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
12883: pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int,
12884: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFnFontEnumProc; p4 : LPARAM ) : BOOL
12885: //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL;
12886: //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fnTermPrc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12887: //Function wEnumICMPProfiles( DC : HDC; ICMPProc : TFNICMEnumProc; p3 : LPARAM ) : Integer
12888: //Function wExtTextOut(dc:HDC,X,Y:Int;Options:Longint;Rect:pRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12889: //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs ) : BOOL
12890: //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts ) : BOOL
12891: //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths ) : BOOL
12892: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths ) : BOOL
12893: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12894: Function wGetEnhMetaFile( p1 : PKOLChar ) : HENHMETAFILE
12895: Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar ) : UINT
12896: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD ) : DWORD
12897: //Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:PGlyphMetrics; cbBuffer : DWORD,
12898: lpvBuffer : Pointer; const lpmat2 : TMat2 ) : DWORD
12899: Function wGetICMPProfile( DC : HDC; var Size : DWORD; Name : PKOLChar ) : BOOL
12900: Function wGetMetaFile( p1 : PKOLChar ) : HMETAFILE
12901: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer ) : Integer
12902: //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer ) : UINT
12903: //Function wGetTextExtentExPoint( DC:HDC;p2:PKOLChar;p3,p4:Integer;p5,p6:PInteger;var p7:TSIZE):BOOL
12904: Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSIZE ) : BOOL
12905: Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSIZE ) : BOOL
12906: Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar ) : Integer
12907: //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric ) : BOOL
12908: Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer ) : BOOL
12909: Function wRemoveFontResource( FileName : PKOLChar ) : BOOL
12910: //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : BOOL
12911: //Function wResetDC( DC : HDC; const InitData : TDeviceMode ) : HDC
12912: Function wSetICMProfile( DC : HDC; Name : PKOLChar ) : BOOL
12913: //Function wStartDoc( DC : HDC; const p2 : TDocInfo ) : Integer
12914: Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer ) : BOOL
12915: Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT ) : BOOL
12916: Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD ) : BOOL
12917: //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12918: Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12919: Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer ) : BOOL
12920: //Function
12921: wCallWindowProc(lpPrevWndFunc:TFNWndProc,hWnd:HWND,Msg:UINT;wParam:WPARAM,lParam:LPARAM):LRESULT
12922: //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD ) : Longint
12923: Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12924: Function wCharLower( lpsz : PKOLChar ) : PKOLChar
12925: Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12926: Function wCharNext( lpsz : PKOLChar ) : PKOLChar
12927: //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12928: Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar ) : PKOLChar
12929: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12930: Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12931: Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12932: Function wCharUpper( lpsz : PKOLChar ) : PKOLChar
12933: Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12934: Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer ) : Integer
12935: Function wCreateAcceleratorTable( var Accel, Count : Integer ) : HACCEL
12936: //Function wCreateDesktop(lpszDesktop,
12937: lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK

```

```

12937: //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
12938:   HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND
12939: //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
12940:   lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND
12941: Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
12942:   hWndParent : HWND; hInstance : HINST; lParam : LPARAM ) : HWND
12943: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
12944:   nWidth, nHeight:Int WndParent:HWND;lMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12945: //Function wCreateWindowStation(lpWinsta:PKOLChar,dwReserv,
12946:   dwDesiredAccess:DWORD;lpsa:PSecurityAttrs):HWINSTA;
12947: Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12948: Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12949: Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT;
12950: Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12951: //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
12952:   : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12953: //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
12954:   : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12955: Function wDispatchMessage( const lpMsg : TMsg ) : Longint
12956: Function wDlgDirList( hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
nIDStaticPath:Integer;uFileType:UINT):Integer;
12957: Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
nIDStaticPath:Int;ufiletype:UINT):Int;
12958: Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer) : BOOL
12959: Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer ) : BOOL
12960: //FuncwDrawState(dc:HDC;Brush:HBRUSH,CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
12961:   cy:Int;Flags:UINT):BOOL;
12962: Function wDrawText(hdc:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12963: Function wFindWindow( lpClassName, lpWindowName : PKOLChar ) : HWND
12964: Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar ) : HWND
12965: //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
12966:   pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12967: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass ) : BOOL
12968: //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx ) : BOOL
12969: Function wGetClassLong( hWnd : HWND; nIndex : Integer ) : DWORD
12970: Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12971: Function wGetDlgItemText( hDlg : HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12972: Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer ) : Integer
12973: Function wGetKeyboardLayoutName( pwszKLID : PKOLChar ) : BOOL
12974: //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo ) : BOOL
12975: Function wGetMenuItemString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12976: Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT ) : BOOL
12977: Function wGetProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12978: //Function wGetTabbedTextExtent( hdc : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
12979:   lpnTabStopPositions ) : DWORD
12980: //Function wGetObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
12981:   lpnLengthNeed:DWORD)BOOL;
12982: Function wGetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
12983: Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT ) : UINT
12984: Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer ) : Integer
12985: Function wGetWindowTextLength( hWnd : HWND ) : Integer
12986: //Function wGrayString(hdc:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnT,X,Y,nWidt,
12987:   nHeigt:Int):BOOL;
12988: Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12989: //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
12990: Function wIsCharAlpha( ch : KOLChar ) : BOOL
12991: Function wIsCharAlphaNumeric( ch : KOLChar ) : BOOL
12992: Function wIsCharLower( ch : KOLChar ) : BOOL
12993: Function wIsCharUpper( ch : KOLChar ) : BOOL
12994: Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg ) : BOOL
12995: Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar ) : HACCEL
12996: Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar ) : HBITMAP
12997: Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar ) : HCURSOR
12998: Function wLoadCursorFromFile( lpFileName : PKOLChar ) : HCURSOR
12999: Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar ) : HICON
13000: Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
13001: Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT ) : HKL
13002: Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar ) : HMENU
13003: //Function wLoadMenuItemIndirect( lpMenuTemplate : Pointer ) : HMENU
13004: Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer : PKOLChar;nBufferMax:Integer):Integer
13005: Function wMapVirtualKey( uCode, uMapType : UINT ) : UINT
13006: Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL ) : UINT
13007: Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT ) : Integer
13008: Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
13009: //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams ) : BOOL
13010: Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
13011: //7Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR ) : BOOL
13012: //Function wOemToBuf( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD ) : BOOL
13013: Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
13014: Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
13015: Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
13016: Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD ) : HWINSTA
13017: Function wPeekMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT ):BOOL
13018: Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13019: Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13020: Function wRealGetWindowClass( hWnd : HWND; pszType : PKOLChar; cchType : UINT ) : UINT
13021: // Function wRegisterClass( const lpWndClass : TWndClass ) : ATOM
13022: // Function wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM

```

```

13012: Function wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
13013: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13014: Function wRegisterWindowMessage( lpString : PKOLChar ) : UINT
13015: Function wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THandle
13016: Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13017: Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
13018: //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
1lpResultCallBack : TFNSendAsynchProc; dwData : DWORD ) : BOOL
13019: Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
1lpdwResult:DWORD) : LRESULT
13020: Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13021: Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
13022: Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar ) : BOOL
13023: //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
13024: Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle ) : BOOL
13025: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
13026: Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint
13027: Function wSetWindowText( hWnd : HWND; lpString : PKOLChar ) : BOOL
13028: //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc ) : HHOOK
13029: //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13030: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni : UINT ):BOOL
13031: Function wTabbedTextOut(hdc:HDC;x,y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
1lpnTabStopPositions,nTabOrigin:Int):Longint;
13032: Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg ) : Integer
13033: Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST ) : BOOL
13034: Function wVKeyScan( ch : KOLChar ) : SHORT
13035: Function wVKeyScanEx( ch : KOLChar; dwhkl : HKL ) : SHORT
13036: Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD ) : BOOL
13037: Function wwsprintf( Output : PKOLChar; Format : PKOLChar ) : Integer
13038: Function wwwsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list ) : Integer
13039:
13040: //TestDrive!
13041: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13042: 'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString('ConvertSidToStringSidA
13043: Function GetDomainUserSids(const domainName:String;const userName:String; var foundDomain:String):String;
13044: Function GetLocalUserSidStr( const UserName : string ) : string
13045: Function getPid4User( const domain : string; const user : string; var pid : dword ) : boolean
13046: Function Impersonate2User( const domain : string; const user : string ) : boolean
13047: Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString ) : Boolean
13048: Function KillProcessByName( const exename : string; var found : integer ) : integer
13049: Function getWinProcessList : TStringList
13050: function WaitTilClose(hWnd: Integer): Integer;
13051: function DoUserMsgs: Boolean;
13052: function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13053: procedure ShowMsg(hParent: Integer; const Mess, Title: String); //modal but NOT blockable
13054: procedure DeleteMsgForm(Handle: Integer);
13055: procedure DisableForms;
13056: function FoundTopLevel(hWnd, LParam: Integer): BOOL; stdCall;
13057: end;
13058:
13059: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13060: begin
13061: 'AfMaxSyncSlots','LongInt'( 64 );
13062: 'AfSynchronizeTimeout','LongInt'( 2000 );
13063: TAfSyncSlotID', 'DWORD
13064: TAfSyncStatistics', 'record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13065: TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID )
13066: TAfSafeDirectSyncEvent', 'Procedure
13067: Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent ) : TAfSyncSlotID
13068: Function AfReleaseSyncSlot( const ID : TAfSyncSlotID ) : Boolean
13069: Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean ) : Boolean
13070: Function AfValidateSyncSlot( const ID : TAfSyncSlotID ) : Boolean
13071: Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD ) : Boolean
13072: Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD ) : Boolean
13073: Function AfIsSyncMethod : Boolean
13074: Function AfSyncWnd : HWnd
13075: Function AfSyncStatistics : TAfSyncStatistics
13076: Procedure AfClearSyncStatistics
13077: end;
13078:
13079: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13080: begin
13081: 'fBinary','LongWord')($00000001);
13082: 'fParity','LongWord')($00000002);
13083: 'fOutxCtsFlow','LongWord').SetUInt($00000004);
13084: 'fOutxDsrFlow','LongWord')($00000008);
13085: 'fDtrControl','LongWord')($00000030);
13086: 'fDtrControlDisable','LongWord')($00000000);
13087: 'fDtrControlEnable','LongWord')($00000010);
13088: 'fDtrControlHandshake','LongWord')($00000020);
13089: 'fDsrSensitivity','LongWord')($00000040);
13090: 'fTXContinueOnXoff','LongWord')($00000080);
13091: 'fOutX','LongWord')($00000100);
13092: 'fInX','LongWord')($00000200);
13093: 'fErrorChar','LongWord')($00000400);
13094: 'fNull','LongWord')($00000800);
13095: 'fRtsControl','LongWord')($00000300);
13096: 'fRtsControlDisable','LongWord')($00000000);
13097: 'fRtsControlEnable','LongWord')($00000100);

```

```

13098: 'fRtsControlHandshake', 'LongWord')( $00002000);
13099: 'fRtsControlToggle', 'LongWord')( $00003000);
13100: 'fAbortOnError', 'LongWord')( $00004000);
13101: 'fDummy2', 'LongWord')( $FFF8000);
13102: TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13103: FindClass('TOBJECT'), 'EAFComPortCoreError
13104: FindClass('TOBJECT'), 'TafComPortCore
13105: TafComPortCoreEvent', 'Procedure ( Sender : TafComPortCore; Even'
13106: +'tKind : TAfCoreEvent; Data : DWORD)
13107: SIRegister_TAfComPortCoreThread(CL);
13108: SIRegister_TAfComPortEventThread(CL);
13109: SIRegister_TAfComPortWriteThread(CL);
13110: SIRegister_TAfComPortCore(CL);
13111: Function FormatDeviceName( PortNumber : Integer ) : string
13112: end;
13113:
13114: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13115: begin
13116:   TAFIFileStreamEvent', 'Function ( const fileName : String; mode: Word ) : TStream
13117:   TAFIFileStreamExistsEvent', 'Function ( const fileName : String ) : Boolean
13118:   SIRegister_TApplicationFileIO(CL);
13119:   TDataFileCapability', '( dfcRead, dfcWrite )
13120:   TDataFileCapabilities', 'set of TDataFileCapability
13121:   SIRegister_TDataFile(CL);
13122: //TDataFileClass', 'class of TDataFile
13123: Function ApplicationFileIODefined : Boolean
13124: Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13125: Function FileStreamExists(const fileName: String) : Boolean
13126: //Procedure Register
13127: end;
13128:
13129: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13130: begin
13131:   TALFBXFieldType', '( uftUnknown, uftNumeric, uftChar, uftVarchar'
13132:   '+, uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecision'
13133:   '+on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13134:   TALFBXScale', 'Integer
13135:   FindClass('TOBJECT'), 'EALFBXConvertError
13136:   SIRegister_EALFBXError(CL);
13137:   SIRegister_EALFBXException(CL);
13138:   FindClass('TOBJECT'), 'EALFBXGFixError
13139:   FindClass('TOBJECT'), 'EALFBXDSQLError
13140:   FindClass('TOBJECT'), 'EALFBXDynError
13141:   FindClass('TOBJECT'), 'EALFBXBakError
13142:   FindClass('TOBJECT'), 'EALFBXGSecError
13143:   FindClass('TOBJECT'), 'EALFBXLicenseError
13144:   FindClass('TOBJECT'), 'EALFBXGStatError
13145: //EALFBXExceptionClass', 'class of EALFBXError
13146: TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13147:   '+37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13148:   '+csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csSOC'
13149:   '+TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252, '
13150:   '+ csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13151:   '+sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13152:   '+_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13153:   '+8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13154: TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13155:   +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13156:   +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13157:   +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13158: TALFBXTransParams', 'set of TALFBXTransParam
13159: Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString ) : TALFBXCharacterSet
13160: Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13161: Function ALFBXCreatetBlobParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13162: 'cALFBXMaxParamLength', 'LongInt'( 125 );
13163: TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13164: TALFBXParamsFlags', 'set of TALFBXParamsFlag
13165: //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13166: //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13167: TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete, '
13168:   +'stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommis'
13169:   +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13170: SIRegister_TALFBXSQLDA(CL);
13171: //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13172: SIRegister_TALFBXPoolStream(CL);
13173: //PALFBXBlobData', '^TALFBXBlobData // will not work
13174: TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13175: //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13176: //TALFBXArrayDesc', 'TISCArrayDesc
13177: //TALFBXBlobDesc', 'TISCblobDesc
13178: //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13179: //TALFBXArrayIndex', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13180: SIRegister_TALFBXSQLResult(CL);
13181: //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13182: SIRegister_TALFBXSQLParams(CL);
13183: //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13184: TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13185:   +'atementType : TALFBXStatementType; end
13186: FindClass('TOBJECT'), 'TALFBXLibrary

```

```

13187: //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13188: TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary )
13189: //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13190: //+' Excep : EALFBXExceptionClass)
13191: SIRegister_TALFBXLibrary(CL);
13192: 'cALFBXDateOffset', 'LongInt'( 15018);
13193: 'cALFBXTimeCoeff', 'LongInt'( 864000000);
13194: //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13195: //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13196: //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13197: Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13198: Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13199: //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13200: //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13201: //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13202: Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13203: Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13204: TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prigno )
13205: TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13206: Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13207: Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13208: end;
13209:
13210: procedure SIRegister_ALFBXClient(CL: TPSPPascalCompiler);
13211: begin
13212:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13213:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13214:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13215:     +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13216:     +'teger; First : Integer; CacheThreshold : Integer; end
13217:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13218:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13219:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13220:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13221:     +'_writes : int64; page_fetches : int64; page_marks : int64; end
13222:   SIRegister_TALFBXClient(CL);
13223:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13224:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13225:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13226:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13227:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13228:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13229:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13230:   SIRegister_TALFBXConnectionPoolClient(CL);
13231:   SIRegister_TALFBXEventThread(CL);
13232:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13233: end;
13234:
13235: procedure SIRegister_ovcBidi(CL: TPSPPascalCompiler);
13236: begin
13237:   _OSVERSIONINFOA = record
13238:     dwOSVersionInfoSize: DWORD;
13239:     dwMajorVersion: DWORD;
13240:     dwMinorVersion: DWORD;
13241:     dwBuildNumber: DWORD;
13242:     dwPlatformId: DWORD;
13243:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13244:   end;
13245:   TOSVersionInfoA', '_OSVERSIONINFOA
13246:   TOSVersionInfo', 'TOSVersionInfoA
13247:   'WS_EX_RIGHT', 'LongWord')($00001000);
13248:   'WS_EX_LEFT', 'LongWord')($00000000);
13249:   'WS_EX_RTLREADING', 'LongWord')($00002000);
13250:   'WS_EX_LTRREADING', 'LongWord')($00000000);
13251:   'WS_EX_LEFTSCROLLBAR', 'LongWord')($00004000);
13252:   'WS_EX_RIGHTSCROLLBAR', 'LongWord')($00000000);
13253:   Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13254:   'LAYOUTRTL', 'LongWord')($00000001);
13255:   'LAYOUT_BTT', 'LongWord')($00000002);
13256:   'LAYOUT_VBH', 'LongWord')($00000004);
13257:   'LAYOUT_BITMAPORIENTATIONPRESERVED', 'LongWord')($00000008);
13258:   'NOMIRRORBITMAP', 'LongWord')((DWORD($80000000)));
13259:   Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13260:   Function GetLayout( dc : hdc) : DWORD
13261:   Function IsBidi : Boolean
13262:   Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13263:   Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13264:   Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13265:   Function GetPriorityClass( hProcess : THandle) : DWORD
13266:   Function OpenClipboard( hWndNewOwner : HWnd) : BOOL
13267:   Function CloseClipboard : BOOL
13268:   Function GetClipboardSequenceNumber : DWORD
13269:   Function GetClipboardOwner : HWnd
13270:   Function SetClipboardViewer( hWndNewViewer : HWnd) : HWnd
13271:   Function GetClipboardViewer : HWnd
13272:   Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWnd) : BOOL
13273:   Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13274:   Function GetClipboardData( uFormat : UINT) : THandle
13275:   Function RegisterClipboardFormat( lpszFormat : PChar) : UINT

```

```

13276: Function CountClipboardFormats : Integer
13277: Function EnumClipboardFormats( format : UINT ) : UINT
13278: Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;/cchMaxCount:Integer):Integer
13279: Function EmptyClipboard : BOOL
13280: Function IsClipboardFormatAvailable( format : UINT ) : BOOL
13281: Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer ) : Integer
13282: Function GetOpenClipboardWindow : HWND
13283: Function EndDialog( hDlg : HWND; nResult : Integer ) : BOOL
13284: Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer ) : HWND
13285: Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13286: Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;/var lpTranslated:BOOL;bSigned: BOOL): UINT
13287: Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar ) : BOOL
13288: Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT ) : BOOL
13289: Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer ) : BOOL
13290: Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer ) : UINT
13291: Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13292: end;
13293:
13294: procedure SIRegister_DXPUtils(CL: TPSPPascalCompiler);
13295: begin
13296:   Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13297:   Function GetTemporaryFilesPath : String
13298:   Function GetTemporaryFileName : String
13299:   Function FindFileInPaths( const fileName, paths : String ) : String
13300:   Function PathsToString( const paths : TStrings ) : String
13301:   Procedure StringToPaths( const pathsString : String; paths : TStrings )
13302: //Function MacroExpandPath( const aPath : String ) : String
13303: end;
13304:
13305: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPPascalCompiler);
13306: begin
13307:   SIRegister_TALMultiPartBaseContent(CL);
13308:   SIRegister_TALMultiPartBaseContents(CL);
13309:   SIRegister_TALMultiPartBaseStream(CL);
13310:   SIRegister_TALMultiPartBaseEncoder(CL);
13311:   SIRegister_TALMultiPartBaseDecoder(CL);
13312:   Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString ) : AnsiString
13313:   Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13314:   Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13315: end;
13316:
13317: procedure SIRegister_SmallUtils(CL: TPSPPascalCompiler);
13318: begin
13319:   TdriveSize', 'record FreeS : Int64; Totals : Int64; end
13320:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In' +
13321:     +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13322:   Function aAllocPadedMem( Size : Cardinal ) : TObject
13323:   Procedure aFreePadedMem( var P : TObject );
13324:   Procedure aFreePadedMem( var P : PChar );
13325:   Function aCheckPadedMem( P : Pointer ) : Byte
13326:   Function aGetPadMemSize( P : Pointer ) : Cardinal
13327:   Function aAllocMem( Size : Cardinal ) : Pointer
13328:   Function aStrLen( const Str : PChar ) : Cardinal
13329:   Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal ) : PChar
13330:   Function aStrECopy( Dest : PChar; const Source : PChar ) : PChar
13331:   Function aStrCopy( Dest : PChar; const Source : PChar ) : PChar
13332:   Function aStrEnd( const Str : PChar ) : PChar
13333:   Function aStrScan( const Str : PChar; aChr : Char ) : PChar
13334:   Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal ) : PChar
13335:   Function aPCharLength( const Str : PChar ) : Cardinal
13336:   Function aPCharUpper( Str : PChar ) : PChar
13337:   Function aPCharLower( Str : PChar ) : PChar
13338:   Function aStrCat( Dest : PChar; const Source : PChar ) : PChar
13339:   Function aLastDelimiter( const Delimiters, S : String ) : Integer
13340:   Function aCopyTail( const S : String; Len : Integer ) : String
13341:   Function aInt2Thos( I : Int64 ) : String
13342:   Function aUpperCase( const S : String ) : String
13343:   Function aLowerCase( const S : string ) : String
13344:   Function aCompareText( const S1, S2 : string ) : Integer
13345:   Function aSameText( const S1, S2 : string ) : Boolean
13346:   Function aInt2Str( Value : Int64 ) : String
13347:   Function aStr2Int( const Value : String ) : Int64
13348:   Function aStr2IntDef( const S : string; Default : Int64 ) : Int64
13349:   Function aGetFileExt( const FileName : String ) : String
13350:   Function aGetFilePath( const FileName : String ) : String
13351:   Function aGetFileName( const FileName : String ) : String
13352:   Function aChangeExt( const FileName, Extension : String ) : String
13353:   Function aAdjustLineBreaks( const S : string ) : string
13354:   Function aGetWindowStr( WinHandle : HWND ) : String
13355:   Function aDiskSpace( Drive : String ) : TdriveSize
13356:   Function aFileExists( FileName : String ) : Boolean
13357:   Function aFileSize( FileName : String ) : Int64
13358:   Function aDirectoryExists( const Name : string ) : Boolean
13359:   Function aSysErrorMessage( ErrorCode : Integer ) : string
13360:   Function aShortPathName( const LongName : string ) : string
13361:   Function aGetWindowVer : TWinVerRec
13362:   procedure InitDriveSpacePtr;
13363: end;
13364:

```

```

13365: procedure SIRегистер_MakeApp(CL: TPSPPascalCompiler);
13366: begin
13367:   aZero', 'LongInt'( 0 );
13368:   'makeappDEF', 'LongInt'( - 1 );
13369:   'CS_VREDRAW', 'LongInt'( DWORD( 1 ) );
13370:   'CS_HREDRAW', 'LongInt'( DWORD( 2 ) );
13371:   'CS_KEYCVTWINDOW', 'LongInt'( 4 );
13372:   'CS_DBLCLKS', 'LongInt'( 8 );
13373:   'CS_OWNDC', 'LongWord')( $20 );
13374:   'CS_CLASSDC', 'LongWord')( $40 );
13375:   'CS_PARENTDC', 'LongWord')( $80 );
13376:   'CS_NOKEYCWT', 'LongWord')( $100 );
13377:   'CS_NOCLOSE', 'LongWord')( $200 );
13378:   'CS_SAVEBITS', 'LongWord')( $800 );
13379:   'CS_BYTEALIGNCLIENT', 'LongWord')( $1000 );
13380:   'CS_BYTEALIGNWINDOW', 'LongWord')( $2000 );
13381:   'CS_GLOBALCLASS', 'LongWord')( $4000 );
13382:   'CS_IME', 'LongWord')( $10000 );
13383:   'CS_DROPSHADOW', 'LongWord')( $20000 );
13384:   //TPanelFunc // will not work
13385:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13386:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13387:   TFontLooks', 'set of TFontLook
13388:   TMessagefunc', 'function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13389:   Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13390:   Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13391:   Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer ) : Word
13392:   Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13393:   Procedure RunMsgLoop( Show : Boolean )
13394:   Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13395:   Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
ID_Number:Cardinal;hFont:Int):Int
13396:   Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13397:   Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13398:   Function MakePanel(Left,Top,Width,Height,
hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int
13399:   Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13400:   Function id4menu( a, b : Byte; c : Byte; d : Byte ) : Cardinal
13401:   Procedure DoInitMakeApp //set first to init formclasscontrol!
13402: end;
13403;
13404: procedure SIRегистер_ScreenSaver(CL: TPSPPascalCompiler);
13405: begin
13406:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13407:     + 'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13408:   TScreenSaverOptions', 'set of TScreenSaverOption
13409:   'cDefaultScreenSaverOptions', 'LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13410:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND )
13411:   SIRегистер_TScreenSaver(CL);
13412:   //Procedure Register
13413:   Procedure SetScreenSaverPassword
13414: end;
13415;
13416: procedure SIRегистер_XCollection(CL: TPSPPascalCompiler);
13417: begin
13418:   FindClass('TOBJECT'), 'TXCollection
13419:   SIRегистер_EFilerException(CL);
13420:   SIRегистер_TXCollectionItem(CL);
13421:   //TXCollectionItemClass', 'class of TXCollectionItem
13422:   SIRегистер_TXCollection(CL);
13423:   Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent )
13424:   Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent )
13425:   Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass )
13426:   Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass )
13427:   Function FindXCollectionItemClass( const className : String ) : TXCollectionItemClass
13428:   Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass ) : TList
13429: end;
13430;
13431: procedure SIRегистер_XOpenGL(CL: TPSPPascalCompiler);
13432: begin
13433:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond, mtcmArbitrary);
13434:   Procedure xglMapTexCoordToNull
13435:   Procedure xglMapTexCoordToMain
13436:   Procedure xglMapTexCoordToSecond
13437:   Procedure xglMapTexCoordToDual
13438:   Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal );
13439:   Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal );
13440:   Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal )
13441:   Procedure xglBeginUpdate
13442:   Procedure xglEndUpdate
13443:   Procedure xglPushState
13444:   Procedure xglPopState
13445:   Procedure xglForbidSecondTextureUnit
13446:   Procedure xglAllowSecondTextureUnit
13447:   Function xglGetBitWiseMapping : Cardinal
13448: end;
13449;
13450: procedure SIRегистер_VectorLists(CL: TPSPPascalCompiler);

```

```

13451: begin
13452:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13453:   TBaseListOptions', 'set of TBaseListOption
13454:   SIRegister_TBaseList(CL);
13455:   SIRegister_TBaseVectorList(CL);
13456:   SIRegister_TAffineVectorList(CL);
13457:   SIRegister_TVectorList(CL);
13458:   SIRegister_TTexPointList(CL);
13459:   SIRegister_TXIntegerList(CL);
13460: //PSingleArrayList', '^TSingleArrayList // will not work
13461:   SIRegister_TSingleList(CL);
13462:   SIRegister_TByteList(CL);
13463:   SIRegister_TQuaternionList(CL);
13464: Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList );
13465: Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList );
13466: Procedure FastQuickSortLists(startIndex,
endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13467: end;
13468:
13469: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13470: begin
13471:   Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList );
13472:   Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList );
13473:   Procedure ConvertStripToList2(const strip:TAffineVectorList;const
indices:TIntegerList;list:TAffineVectorList);
13474:   Procedure ConvertIndexedListToList( const data:TAffineVectlist;const
indices:TIntegerList;list:TAffineVectorList );
13475:   Function BuildVectorCountOptimizedIndices( const vertices:TAffineVectorList; const
normals:TAffineVectorList; const texcoords : TAffineVectorList ) : TIntegerList;
13476:   Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList );
13477:   Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList );
13478:   Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList );
13479:   Function RemapIndicesToIndicesMap( remapIndices : TIntegerList ) : TIntegerList
13480:   Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList )
13481:   Procedure RemapIndices( indices, indicesMap : TIntegerList )
13482:   Procedure UnifyTrianglesWinding( indices : TIntegerList )
13483:   Procedure InvertTrianglesWinding( indices : TIntegerList )
13484:   Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList ) : TAffineVectorList
13485:   Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
edgesTriangles : TIntegerList ) : TIntegerList
13486:   Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single )
13487:   Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):TPersistentObjectList;
13488:   Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer )
13489:   Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices : TIntegerList;
normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent )
13490: end;
13491:
13492: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13493: begin
13494:   Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte )
13495:   Procedure FreeMemAndNil( var P : TObject )
13496:   Function PCharOrNil( const S : string ) : PChar
13497:   SIRegister_TJclReferenceMemoryStream(CL);
13498:   FindClass('TOBJECT'),'EJclVMTError
13499:   {Function GetVirtualMethodCount( AClass : TClass ) : Integer
13500:   Function GetVirtualMethod( AClass : TClass; const Index : Integer ) : Pointer
13501:   Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer )
13502:   PDynamicIndexList', '^TDynamicIndexList // will not work
13503:   PDynamicAddressList', '^TDynamicAddressList // will not work
13504:   Function GetDynamicMethodCount( AClass : TClass ) : Integer
13505:   Function GetDynamicIndexList( AClass : TClass ) : PDYNAMICIndexList
13506:   Function GetDynamicAddressList( AClass : TClass ) : PDYNAMICAddressList
13507:   Function HasDynamicMethod( AClass : TClass; Index : Integer ) : Boolean
13508:   Function GetDynamicMethod( AClass : TClass; Index : Integer ) : Pointer
13509:   Function GetInitTable( AClass : TClass ) : PTTypeInfo
13510:   PFieldEntry', '^TFieldEntry // will not work)
13511:   TFieldEntry', 'record Offset : Integer; IDX : Word; Name : ShortString; end
13512:   Function JIsClass( Address : Pointer ) : Boolean
13513:   Function JIsObject( Address : Pointer ) : Boolean
13514:   Function GetImplementorOfInterface( const I : IInterface ) : TObject
13515:   TDigitCount', 'Integer
13516:   SIRegister_TJclNumericFormat(CL);
13517:   Function JIntToStrZeroPad( Value, Count : Integer ) : AnsiString
13518:   TTextHandler', 'Procedure ( const Text : string )
13519: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223 );
13520:   Function JExecute(const
CommandLine:string;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Cardinal;
13521:   Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13522:   Function ReadKey : Char //to and from the DOS console !
13523:   TModuleHandle', 'HINST
13524:   //TModuleHandle', 'Pointer
13525:   'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ) );
13526:   Function LoadModule( var Module : TModuleHandle; FileName : string ) : Boolean
13527:   Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal ) : Boolean
13528:   Procedure UnloadModule( var Module : TModuleHandle )
13529:   Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string ) : Pointer
13530:   Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean ) : Pointer
13531:   Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;

```

```

13532: Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13533:   FindClass('TOBJECT'),'EJclConversionError'
13534: Function JStrToBoolean( const S : string ) : Boolean
13535: Function JBooleanToStr( B : Boolean ) : string
13536: Function JIntToBool( I : Integer ) : Boolean
13537: Function JBoolToInt( B : Boolean ) : Integer
13538: 'ListSeparator','String ';
13539: 'ListSeparator1','String ';
13540: Procedure ListAddItems( var List : string; const Separator, Items : string)
13541: Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13542: Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13543: Procedure ListDeleteItem( var List : string; const Separator : string; const Index : Integer)
13544: Function ListItemCount( const List, Separator : string ) : Integer
13545: Function ListItem( const List, Separator : string; const Index : Integer ) : string
13546: Procedure ListItemIndex(var List:string;const Separator:string;const Index:Integer;const Value:string)
13547: Function ListItemIndex( const List, Separator, Item : string ) : Integer
13548: Function SystemToObjectInstance : LongWord
13549: Function IsCompiledWithPackages : Boolean
13550: Function JJclGUIDToString( const GUID : TGUID ) : string
13551: Function JJclStringToGUID( const S : string ) : TGUID
13552: SIRegister_TJclIntfCriticalSection(CL);
13553: SIRegister_TJclSimpleLog(CL);
13554: Procedure InitSimpleLog( const ALogFileFileName : string )
13555: end;
13556:
13557: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13558: begin
13559:   FindClass('TOBJECT'),'EJclBorRADException
13560:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13561:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13562:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13563:   TJclBorRADToolPath', 'string
13564: 'SupportedDelphiVersions','LongInt'( 5 or 6 or 7 or 8 or 9 or 10 or 11);
13565: 'SupportedBCBVersions','LongInt'( 5 or 6 or 10 or 11);
13566: 'SupportedBDSVersions','LongInt'( 1 or 2 or 3 or 4 or 5);
13567: BorRADToolRepositoryPagesSection','String 'Repository Pages
13568: BorRADToolRepositoryDialogsPage','String 'Dialogs
13569: BorRADToolRepositoryFormsPage','String 'Forms
13570: BorRADToolRepositoryProjectsPage','String 'Projects
13571: BorRADToolRepositoryDataModulesPage','String 'Data Modules
13572: BorRADToolRepositoryObjectType','String 'Type
13573: BorRADToolRepositoryFormTemplate','String 'FormTemplate
13574: BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13575: BorRADToolRepositoryObjectName','String 'Name
13576: BorRADToolRepositoryObjectPage','String 'Page
13577: BorRADToolRepositoryObjectIcon','String 'Icon
13578: BorRADToolRepositoryObjectDescr','String 'Description
13579: BorRADToolRepositoryObjectAuthor','String 'Author
13580: BorRADToolRepositoryObjectAncestor','String 'Ancestor
13581: BorRADToolRepositoryObjectDesigner','String 'Designer
13582: BorRADToolRepositoryDesignerDfm','String 'dfm
13583: BorRADToolRepositoryDesignerXfm','String 'xmf
13584: BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13585: BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13586: SourceExtensionDelphiPackage','String '.dpk
13587: SourceExtensionBCBPackage','String '.bpk
13588: SourceExtensionDelphiProject','String '.dpr
13589: SourceExtensionBCBProject','String '.bpr
13590: SourceExtensionBDSPackage','String '.bdsproj
13591: SourceExtensionDProject','String '.dproj
13592: BinaryExtensionPackage','String '.bpl
13593: BinaryExtensionLibrary','String '.dll
13594: BinaryExtensionExecutable','String '.exe
13595: CompilerExtensionDCP','String '.dcp
13596: CompilerExtensionBPI','String '.bpi
13597: CompilerExtensionLIB','String '.lib
13598: CompilerExtensionTDS','String '.tds
13599: CompilerExtensionMAP','String '.map
13600: CompilerExtensionDRC','String '.drc
13601: CompilerExtensionDEF','String '.def
13602: SourceExtensionCPP','String '.cpp
13603: SourceExtensionH','String '.h
13604: SourceExtensionPAS','String '.pas
13605: SourceExtensionDFM','String '.dfm
13606: SourceExtensionXFM','String '.xfm
13607: SourceDescriptionPAS','String 'Pascal source file
13608: SourceDescriptionCPP','String 'C++ source file
13609: DesignerVCL','String 'VCL
13610: DesignerCLX','String 'CLX
13611: ProjectTypePackage','String 'package
13612: ProjectTypeLibrary','String 'library
13613: ProjectTypeProgram','String 'program
13614: Personality32Bit','String '32 bit
13615: Personality64Bit','String '64 bit
13616: PersonalityDelphi','String 'Delphi
13617: PersonalityDelphiDotNet','String 'Delphi.net
13618: PersonalityBCB','String 'C++Builder
13619: PersonalityCSB','String 'C#Builder
13620: PersonalityVB','String 'Visual Basic

```

```

13621: PersonalityDesign', 'String 'Design
13622: PersonalityUnknown', 'String 'Unknown personality
13623: PersonalityBDS', 'String 'Borland Developer Studio
13624: DOFDirectoriesSection', 'String 'Directories
13625: DOFUUnitOutputDirKey', 'String 'UnitOutputDir
13626: DOFSearchPathName', 'String 'SearchPath
13627: DOFConditionals', 'String 'Conditionals
13628: DOFLinkerSection', 'String 'Linker
13629: DOFPackagesKey', 'String 'Packages
13630: DOFCompilerSection', 'String 'Compiler
13631: DOFPackageNoLinkKey', 'String 'PackageNoLink
13632: DOFAdditionalSection', 'String 'Additional
13633: DOFOptionsKey', 'String 'Options
13634: TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13635: + 'bpBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13636: +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13637: TJclBorPersonalities', 'set of TJclBorPersonality
13638: TJclBorDesigner', '( bdVCL, bdCLX )
13639: TJclBorDesigners', 'set of TJclBorDesigner
13640: TJclBorPlatform', '( bp32bit, bp64bit )
13641: FindClass('TOBJECT'), 'TJclBorRADToolInstallation
13642: SIRegister_TJclBorRADToolInstallationObject(CL);
13643: SIRegister_TJclBorLandOpenHelp(CL);
13644: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13645: TJclHelp2Objects', 'set of TJclHelp2Object
13646: SIRegister_TJclHelp2Manager(CL);
13647: SIRegister_TJclBorRADToolIDETool(CL);
13648: SIRegister_TJclBorRADToolIDEPackages(CL);
13649: SIRegister_IJclCommandLineTool(CL);
13650: FindClass('TOBJECT'), 'EJclCommandLineToolError
13651: SIRegister_TJclCommandLineTool(CL);
13652: SIRegister_TJclBorLandCommandLineTool(CL);
13653: SIRegister_TJclBCC32(CL);
13654: SIRegister_TJclDCC32(CL);
13655: TJclDCC', 'TJclDCC32
13656: SIRegister_TJclBpr2Mak(CL);
13657: SIRegister_TJclBorLandMake(CL);
13658: SIRegister_TJclBorRADToolPalette(CL);
13659: SIRegister_TJclBorRADToolRepository(CL);
13660: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
13661: TCommandLineTools', 'set of TCommandLineTool
13662: //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13663: SIRegister_TJclBorRADToolInstallation(CL);
13664: SIRegister_TJclBCBInstallation(CL);
13665: SIRegister_TJclDelphiInstallation(CL);
13666: SIRegister_TJclDCCIL(CL);
13667: SIRegister_TJclBDSInstallation(CL);
13668: TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation ) : Boolean
13669: SIRegister_TJclBorRADToolInstallations(CL);
13670: Function BPLFileName( const BPLPath, PackageFileName : string ) : string
13671: Function BinaryFileName( const OutputPath, ProjectFileName : string ) : string
13672: Function IsDelphiPackage( const FileName : string ) : Boolean
13673: Function IsDelphiProject( const FileName : string ) : Boolean
13674: Function IsBCBPackage( const FileName : string ) : Boolean
13675: Function IsBCBProject( const FileName : string ) : Boolean
13676: Procedure GetDPRFileInfo( const DPRFileName:string;out BinaryExtensiion:string;const LibSuffix:PString );
13677: Procedure GetBPRFileInfo( const BPRFileName:string;out BinaryFileName:string;const Descript:PString );
13678: Procedure GetDPKFileInfo( const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
Descript:PString;
13679: Procedure GetBPKFileInfo( const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
Descript:PString
13680: function SamePath(const Path1, Path2: string): Boolean;
13681: end;
13682:
13683: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13684: begin
13685: 'ERROR_NO_MORE_FILES', 'LongInt'( 18 );
13686: //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13687: //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64 ) : Integer
13688: //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13689: 'LPathSeparator', 'String '/'
13690: 'LDirDelimiter', 'String '
13691: 'LDirSeparator', 'String :
13692: 'JXPathDevicePrefix', 'String '\\.\\
13693: 'JXPathSeparator', 'String \
13694: 'JXDirDelimiter', 'String \
13695: 'JXDirSeparator', 'String ';
13696: 'JXPathDhcPrefix', 'String \
13697: 'faNormalFile', 'LongWord')($00000080);
13698: //faUnixSpecific', 'faSymLink;
13699: JXTCompactPath', '( cpCenter, cpEnd )
13700: _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13701: +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13702: +'TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13703: TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13704: WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13705:
13706: Function jxPathAddSeparator( const Path : string ) : string
13707: Function jxPathAddExtension( const Path, Extension : string ) : string

```

```

13708: Function jxPathAppend( const Path, Append : string ) : string
13709: Function jxPathBuildRoot( const Drive : Byte ) : string
13710: Function jxPathCanonicalize( const Path : string ) : string
13711: Function jxPathCommonPrefix( const Path1, Path2 : string ) : Integer
13712: Function jxPathCompactPath( const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13713: Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string )
13714: Function jxPathExtractFileDirFixed( const S : string ) : string
13715: Function jxPathExtractFileNameNoExt( const Path : string ) : string
13716: Function jxPathExtractPathDepth( const Path : string; Depth : Integer ) : string
13717: Function jxPathGetDepth( const Path : string ) : Integer
13718: Function jxPathGetLongName( const Path : string ) : string
13719: Function jxPathGetShortName( const Path : string ) : string
13720: Function jxPathGetLongName( const Path : string ) : string
13721: Function jxPathGetShortName( const Path : string ) : string
13722: Function jxPathGetRelativePath( Origin, Destination : string ) : string
13723: Function jxPathGetTempPath : string
13724: Function jxPathIsAbsolute( const Path : string ) : Boolean
13725: Function jxPathIsChild( const Path, Base : string ) : Boolean
13726: Function jxPathIsDiskDevice( const Path : string ) : Boolean
13727: Function jxPathIsUNC( const Path : string ) : Boolean
13728: Function jxPathRemoveSeparator( const Path : string ) : string
13729: Function jxPathRemoveExtension( const Path : string ) : string
13730: Function jxPathGetPhysicalPath( const LocalizedPath : string ) : string
13731: Function jxPathGetLocalizedPath( const PhysicalPath : string ) : string
13732: JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders )
13733: JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13734: TFileHandler', 'Procedure ( const FileName : string )
13735: TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec )
13736: BuildFileDialog( const Path : string; const Attr : Integer; const List : TStrings ) : Boolean
13737: //Function AdvBuildFileDialog( const Path : string; const Attr : Integer; const Files : TStrings; const
13738: AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
13739: FileMatchFunc:TFileMatchFunc):Bool;
13740: Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int ) : Boolean
13741: Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13742: Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13743: Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
13744: RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13745: Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
13746: IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13747: Procedure jxCreateEmptyFile( const FileName : string )
13748: Function jxCloseVolume( var Volume : THandle ) : Boolean
13749: Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean ) : Boolean
13750: Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13751: Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13752: Function jxDelTree( const Path : string ) : Boolean
13753: //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13754: Function jxDiskInDrive( Drive : Char ) : Boolean
13755: Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean ) : Boolean
13756: Function jxFolderCreateTemp( var Prefix : string ) : THandle
13757: Function jxFolderBackup( const FileName : string; Move : Boolean ) : Boolean
13758: Function jxFolderCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13759: Function jxFolderDelete( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
13760: Function jxFolderExists( const FileName : string ) : Boolean
13761: Function jxFolderMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13762: Function jxFolderRestore( const FileName : string ) : Boolean
13763: Function jxFolderGetSize( const FileName : string ) : Int64
13764: Function jxFolderGetTempName( const Prefix : string ) : string
13765: Function jxFolderGetType( const FileName : string ) : string
13766: Function jxFolderGetUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string ) : string
13767: Function jxFolderForceDirectories( Name : string ) : Boolean
13768: Function jxFolderGetDirectoryName( const FileName : string ) : string
13769: Function jxFolderGetGroupName( const FileName : string; ResolveSymLinks : Boolean ) : string
13770: Function jxFolderGetOwnerName( const FileName : string; ResolveSymLinks : Boolean ) : string
13771: Function jxFolderGetSize( const FileName : string ) : Int64
13772: Function jxFolderGetDriveTypeStr( const Drive : Char ) : string
13773: Function jxFolderGetAgeCoherence( const FileName : string ) : Boolean
13774: Procedure jxFolderGetAttributeList( const Items : TStrings; const Attr : Integer )
13775: Procedure jxFolderGetAttributeListEx( const Items : TStrings; const Attr : Integer )
13776: Function jxFolderGetInformation( const FileName : string; out FileInfo : TSearchRec ) : Boolean;
13777: Function jxFolderGetInformation1( const FileName : string ) : TSearchRec;
13778: //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
13779: ResolveSymLinks:Boolean):Integer
13780: Function jxFolderGetLastWrite( const FName : string ) : TFileTime;
13781: Function jxFolderGetLastAccess( const FName : string; out LocalTime : TDateTime ) : Boolean;
13782: Function jxFolderGetLastAccess1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13783: Function jxFolderGetCreation( const FName : string ) : TFileTime;
13784: Function jxFolderGetCreation1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13785: Function jxFolderGetLastWrite1( const FName : string; out TimeStamp:Integer;ResolveSymLinks : Bool ):Bool;
13786: Function jxFolderGetLastWrite1( const FName : string; out LocalTime:TDateTime;ResolveSymLinks:Bool ): Bool;
13787: Function jxFolderGetLastAccess2( const FName : string; ResolveSymLinks : Boolean ) : Integer;
13788: Function jxFolderGetLastAccess( const FName : string; out TimeStamp:Integer;ResolveSymLinks : Bool ) : Bool;
13789: Function jxFolderGetLastAccess1( const FName : string; out LocalTime:TDateTime;ResolveSymLinks:Bool ) : Bool;
13790: Function jxFolderGetLastAccess2( const FName : string; ResolveSymLinks: Boolean ) : Integer;
13791: Function jxFolderGetLastAttrChange( const FName : string; out TimeStamp:Integer;ResolveSymLinks:Bool ) : Bool;

```

```

13792: Function jxGetFileLastAttrChange1( const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool ) : Bool;
13793: Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean) : Integer;
13794: Function jxGetModulePath( const Module : HMODULE ) : string;
13795: Function jxGetSizeOfFile( const FileName : string ) : Int64;
13796: Function jxGetSizeOfFile1( const FileInfo : TSearchRec ) : Int64;
13797: Function jxGetSizeOfFile2( Handle : THandle ) : Int64;
13798: Function jxGetStandardFileInfo( const FileName : string ) : TWin32FileAttributeData;
13799: Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean;
13800: Function jxIsRootDirectory( const CanonicalFileName : string ) : Boolean;
13801: Function jxLockVolume( const Volume : string; var Handle : THandle ) : Boolean;
13802: Function jxOpenVolume( const Drive : Char ) : THandle;
13803: Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime ) : Boolean;
13804: Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime ) : Boolean;
13805: Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime ) : Boolean;
13806: Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime ) : Boolean;
13807: Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime ) : Boolean;
13808: Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime ) : Boolean;
13809: Procedure jxShredFile( const FileName : string; Times : Integer );
13810: Function jxUnlockVolume( var Handle : THandle ) : Boolean;
13811: Function jxCreateSymbolicLink( const Name, Target : string ) : Boolean;
13812: Function jxSymbolicLinkTarget( const Name : string ) : string;
13813: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )';
13814: SIRegister_TJclCustomFileAttrMask(CL);
13815: SIRegister_TJclFileAttributeMask(CL);
13816: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHidden$';
13817: +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)';
13818: TFileSearchOptions', 'set of TFileSearchOption';
13819: TFileSearchTaskID', 'Integer;
13820: TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc';
13821: +'hTaskID; const Aborted : Boolean)';
13822: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )';
13823: SIRegister_IJclFileEnumerator(CL);
13824: SIRegister_TJclFileEnumerator(CL);
13825: Function JxFileSearch : IJclFileEnumerator;
13826: JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched, ffPreRelease, ffPrivateBuild, ffSpecialBuild )';
13827: JxTFileFlags', 'set of TFileFlag;
13828: FindClass('TOBJECT'), 'EJclFileVersionInfoError';
13829: SIRegister_TJclFileVersionInfo(CL);
13830: Function jxOSIdentToString( const OSIdent : DWORD ) : string;
13831: Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD ) : string;
13832: Function jxVersionResourceAvailable( const FileName : string ) : Boolean;
13833: TFileVersionFormat', '( vfMajorMinor, vfFull )';
13834: Function jxFormatVersionString( const HiV, Lov : Word ) : string;
13835: Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word ) : string;
13836: //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo; VersionFormat:TFFileVersionFormat):str;
13837: //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13838: //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
13839: Revision:Word);
13840: //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo ) : Boolean;
13841: Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFFileVersionFormat; const
13842: NotAvailableText : string ) : string;
13843: SIRegister_TJclTempFileStream(CL);
13844: FindClass('TOBJECT'), 'TJclCustomFileMapping';
13845: SIRegister_TJclFileMappingView(CL);
13846: TJclFileMappingRoundOffset', '( rvDown, rvUp )';
13847: SIRegister_TJclCustomFileMapping(CL);
13848: SIRegister_TJclFileMapping(CL);
13849: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )';
13850: //PPCharArray', '^TPCharArray // will not work';
13851: SIRegister_TJclMappedTextReader(CL);
13852: SIRegister_TJclFileMaskComparator(CL);
13853: FindClass('TOBJECT'), 'EJclPathError';
13854: FindClass('TOBJECT'), 'EJclFileUtilsError';
13855: FindClass('TOBJECT'), 'EJclTempFileStreamError';
13856: FindClass('TOBJECT'), 'EJclTempFileStreamError';
13857: FindClass('TOBJECT'), 'EJclFileMappingError';
13858: FindClass('TOBJECT'), 'EJclFileMappingViewError';
13859: Function jxPathGetLongName2( const Path : string ) : string;
13860: Function jxWin32Deletefile( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean;
13861: Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string ) : Boolean;
13862: Function jxWin32BackupFile( const FileName : string; Move : Boolean ) : Boolean;
13863: Function jxWin32RestoreFile( const FileName : string ) : Boolean;
13864: Function jxSamePath( const Path1, Path2 : string ) : Boolean;
13865: Procedure jxPathListAddItems( var List : string; const Items : string );
13866: Procedure jxPathListIncludeItems( var List : string; const Items : string );
13867: Procedure jxPathListDeleteItems( var List : string; const Items : string );
13868: Procedure jxPathListDeleteItem( var List : string; const Index : Integer );
13869: Function jxPathListItemCount( const List : string ) : Integer;
13870: Function jxPathListGetItem( const List : string; const Index : Integer ) : string;
13871: Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string );
13872: Function jxPathListItemIndex( const List, Item : string ) : Integer;
13873: Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13874: Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13875: Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
13876: AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13877: Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
13878: AllowedPrefixCharacters : string ) : Integer;

```

```

13877: end;
13878:
13879: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13880: begin
13881:   'UTF8FileHeader','String #$ef#$bb#$bf';
13882:   Function lCompareFilenames( const Filenamel, Filename2 : string) : integer
13883:   Function lCompareFilenamesIgnoreCase( const Filenamel, Filename2 : string) : integer
13884:   Function lCompareFilenames( const Filenamel, Filename2 : string; ResolveLinks : boolean) : integer
13885:   Function lCompareFilenames(Filenamel:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13886:   Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13887:   Function lFilenameIsWinAbsolute( const Thefilename : string) : boolean
13888:   Function lFilenameIsUnixAbsolute( const Thefilename : string) : boolean
13889:   Procedure lCheckIfFileIsExecutable( const Afilename : string)
13890:   Procedure lCheckIfFileIsSymlink( const AFilename : string)
13891:   Function lFileIsReadable( const AFilename : string) : boolean
13892:   Function lFileIsWritable( const AFilename : string) : boolean
13893:   Function lFileIsText( const AFilename : string) : boolean
13894:   Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13895:   Function lFileIsExecutable( const AFilename : string) : boolean
13896:   Function lFileIsSymlink( const AFilename : string) : boolean
13897:   Function lFileIsHardLink( const AFilename : string) : boolean
13898:   Function lFileSize( const Filename : string) : int64;
13899:   Function lGetFileDescription( const AFilename : string) : string
13900:   Function lReadAllLinks( const Filenam : string; ExceptionOnError : boolean) : string
13901:   Function lTryReadAllLinks( const Filenam : string) : string
13902:   Function lDirPathExists( const FileName : String) : Boolean
13903:   Function lForceDirectory( DirectoryName : string) : boolean
13904:   Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13905:   Function lProgramDirectory : string
13906:   Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13907:   Function lExtractFileNameOnly( const AFilename : string) : string
13908:   Function lExtractFileNameWithoutExt( const AFilename : string) : string
13909:   Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13910:   Function lCompareFileExt( const Filename, Ext : string) : integer;
13911:   Function lFilenameIsPascalUnit( const Filename : string) : boolean
13912:   Function lAppendPathDelim( const Path : string) : string
13913:   Function lChompPathDelim( const Path : string) : string
13914:   Function lTrimFilename( const AFilename : string) : string
13915:   Function lCleanAndExpandFilename( const Filenam : string) : string
13916:   Function lCleanAndExpandDirectory( const Filenam : string) : string
13917:   Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13918:   Function lCreateRelativePath( const Filenam, BaseDirectory : string; UsePointDirectory : boolean;
  AlwaysRequireSharedBaseFolder : Boolean) : string
13919:   Function lCreateAbsolutePath( const Filenam, BaseDirectory : string) : string
13920:   Function lFileIsInPath( const Filenam, Path : string) : boolean
13921:   Function lFileIsInDirectory( const Filenam, Directory : string) : boolean
13922:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13923:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13924:   'AllDirectoryEntriesMask','String '*
13925:   Function l GetAllFilesMask : string
13926:   Function lGetExeExt : string
13927:   Function lSearchFileInPath( const Filenam, basePath, SearchPath, Delimiter : string; Flags : TSearchFileInPathFlags ) : string
13928:   Function lSearchAllFilesInPath( const Filenam, basePath, SearchPath, Delimiter:string;Flags : TSearchFileInPathFlags ) : TStrings
13929:   Function lFindDiskFilename( const Filenam : string) : string
13930:   Function lFindDiskFileCaseInsensitive( const Filenam : string) : string
13931:   Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13932:   Function lGetDarwinSystemFilename( Filenam : string) : string
13933:   SIRegister_TFileIterator(CL);
13934:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13935:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13936:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13937:   SIRegister_TFileSearcher(CL);
13938:   Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13939:   Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13940:   // TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13941:   // TCopyFileFlags', 'set of TCopyFileFlag
13942:   Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13943:   Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13944:   Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13945:   Function lReadFileToString( const Filenam : string) : string
13946:   Function lGetTempFilename( const Directory, Prefix : string) : string
13947:   {Function NeedRTLAnsi : boolean
13948:   Procedure SetNeedRTLAnsi( NewValue : boolean)
13949:   Function UTF8ToSys( const s : string) : string
13950:   Function SysToUTF8( const s : string) : string
13951:   Function ConsoleToUTF8( const s : string) : string
13952:   Function UTF8ToConsole( const s : string) : string
13953:   Function FileExistsUTF8( const Filenam : string) : boolean
13954:   Function FileAgeUTF8( const FileName : string) : Longint
13955:   Function DirectoryExistsUTF8( const Directory : string) : Boolean
13956:   Function ExpandFileNameUTF8( const FileName : string) : string
13957:   Function ExpandUNCFileNameUTF8( const FileName : string) : string
13958:   Function ExtractShortPathNameUTF8( const FileName : String) : String
13959:   Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13960:   Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13961:   Procedure FindCloseUTF8( var F : TSearchrec)
13962:   Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint

```

```

13963: Function FileGetAttrUTF8( const FileName : String ) : Longint
13964: Function FileSetAttrUTF8( const FileName : String; Attr : longint ) : Longint
13965: Function DeleteFileUTF8( const FileName : String ) : Boolean
13966: Function RenameFileUTF8( const OldName, NewName : String ) : Boolean
13967: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean ) : String
13968: Function FileIsReadOnlyUTF8( const FileName : String ) : Boolean
13969: Function GetCurrentDirUTF8 : String
13970: Function SetCurrentDirUTF8( const NewDir : String ) : Boolean
13971: Function CreateDirUTF8( const NewDir : String ) : Boolean
13972: Function RemoveDirUTF8( const Dir : String ) : Boolean
13973: Function ForceDirectoriesUTF8( const Dir : string ) : Boolean
13974: Function FileOpenUTF8( const FileName : string; Mode : Integer ) : THandle
13975: Function FileCreateUTF8( const FileName : string ) : THandle;
13976: Function FileCreateUTF8l( const FileName : string; Rights : Cardinal ) : THandle;
13977: Function ParamStrUTF8( Param : Integer ) : string
13978: Function GetEnvironmentStringUTF8( Index : Integer ) : string
13979: Function GetEnvironmentVariableUTF8( const EnvVar : string ) : String
13980: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean ) : string
13981: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean ) : string
13982: Function SysErrorMessageUTF8( ErrorCode : Integer ) : String
13983: end;
13984:
13985: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13986: begin
13987:   //VK_F23 = 134;
13988:   //{$EXTERNALSYM VK_F24}
13989:   //VK_F24 = 135;
13990:   TVirtualKeyCode', 'Integer
13991:   'VK_MOUSEWHEELUP','integer'(134);
13992:   'VK_MOUSEWHEELDOWN','integer'(135);
13993:   Function glIsKeyDown( c : Char ) : Boolean;
13994:   Function glIsKeyDown1( vk : TVirtualKeyCode ) : Boolean;
13995:   Function glKeyPressed( minVkeyCode : TVirtualKeyCode ) : TVirtualKeyCode
13996:   Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode ) : String
13997:   Function glKeyNameToVirtualKeyCode( const keyName : String ) : TVirtualKeyCode
13998:   Function glCharToVirtualKeyCode( c : Char ) : TVirtualKeyCode
13999:   Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer )
14000: end;
14001:
14002: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14003: begin
14004:   TGLPoint', 'TPoint
14005:   //PGLPoint', '^TGLPoint // will not work
14006:   TGLRect', 'TRect
14007:   //PGLRect', '^TGLRect // will not work
14008:   TDelphiColor', 'TColor
14009:   TGLPicture', 'TPicture
14010:   TGLGraphic', 'TGraphic
14011:   TGLBitmap', 'TBitmap
14012:   //TGraphicClass', 'class of TGraphic
14013:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14014:   TGLMouseEvent', '( mbLeft, mbRight, mbMiddle )
14015:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
14016:     +'Button; Shift : TShiftState; X, Y : Integer)
14017:   TGLMouseMoveEvent', 'TMouseEvent
14018:   TGLKeyEvent', 'TKeyEvent
14019:   TGLKeyPressEvent', 'TKeyPressEvent
14020:   EGLOSError', 'EWin32Error
14021:   EGLOSError', 'EWin32Error
14022:   EGLOSError', 'EOSError
14023:   'glsAllFilter', 'string'All // $AllFilter
14024:   Function GLPoint( const x, y : Integer ) : TGLPoint
14025:   Function GLRGB( const r, g, b : Byte ) : TColor
14026:   Function GLColorToRGB( color : TColor ) : TColor
14027:   Function GLGetRValue( rgb : DWORD ) : Byte
14028:   Function GLGetGValue( rgb : DWORD ) : Byte
14029:   Function GLGetBValue( rgb : DWORD ) : Byte
14030:   Procedure GLInitWinColors
14031:   Function GLRect( const aLeft, aTop, aRight, aBottom : Integer ) : TGLRect
14032:   Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer )
14033:   Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect )
14034:   Procedure GLInformationDlg( const msg : String )
14035:   Function GLQuestionDlg( const msg : String ) : Boolean
14036:   Function GLInputDlg( const aCaption, aPrompt, aDefault : String ) : String
14037:   Function GLSavePictureDialog( var aFileName : String; const aTitle : String ) : Boolean
14038:   Function GLOpenPictureDialog( var aFileName : String; const aTitle : String ) : Boolean
14039:   Function GLApplicationTerminated : Boolean
14040:   Procedure GLRaiseLastOSError
14041:   Procedure GLFreeAndNil( var anObject : TObject )
14042:   Function GLGetDeviceLogicalPixelsX( device : Cardinal ) : Integer
14043:   Function GLGetCurrentColorDepth : Integer
14044:   Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat ) : Integer
14045:   Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer ) : Pointer
14046:   Procedure GLSleep( length : Cardinal )
14047:   Procedure GLQueryPerformanceCounter( var val : Int64 )
14048:   Function GLQueryPerformanceFrequency( var val : Int64 ) : Boolean
14049:   Function GLStartPrecisionTimer : Int64
14050:   Function GLPrecisionTimerLap( const precisionTimer : Int64 ) : Double
14051:   Function GLStopPrecisionTimer( const precisionTimer : Int64 ) : Double

```

```

14052: Function GLRDTSC : Int64
14053: Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14054: Function GLOKMessageBox( const Text, Caption : string) : Integer
14055: Procedure GLShowHTMLUrl( Url : String)
14056: Procedure GLShowCursor( AShow : boolean)
14057: Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14058: Procedure GLGetCursorPos( var point : TGLPoint)
14059: Function GLGetScreenWidth : integer
14060: Function GLGetScreenHeight : integer
14061: Function GLGetTickCount : int64
14062: function RemoveSpaces(const str : String) : String;
14063: TNormalMapSpace'( nmsObject, nmsTangent )
14064: Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList:Colors:TVectorList)
14065: Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14066: Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
14067: TAffineVectorList; Colors : TVectorList)
14068: Function CreateObjectSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
14069: LoTexCoords, Tangents, BiNormals : TAffineVectorList ) : TGLBitmap
14070: end;
14071: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14072: begin
14073:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14074: // PGLStarRecord', '^TGLStarRecord // will not work
14075: Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14076: Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14077: Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single ) : TVector
14078: end;
14079:
14080:
14081: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14082: begin
14083:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14084: //PAABB', '^TAABB // will not work
14085: TBSphere', 'record Center : TAffineVector; Radius : single; end
14086: TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14087: TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14088: Function AddBB( var cl : THmgBoundingBox; const c2 : THmgBoundingBox ) : THmgBoundingBox
14089: Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14090: Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14091: Procedure SetAABB( var bb : TAABB; const v : TVector)
14092: Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14093: Procedure AABBTTransform( var bb : TAABB; const m : TMatrix)
14094: Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14095: Function BBMinX( const c : THmgBoundingBox ) : Single
14096: Function BBMaxX( const c : THmgBoundingBox ) : Single
14097: Function BBMinY( const c : THmgBoundingBox ) : Single
14098: Function BBMaxY( const c : THmgBoundingBox ) : Single
14099: Function BBMinZ( const c : THmgBoundingBox ) : Single
14100: Function BBMaxZ( const c : THmgBoundingBox ) : Single
14101: Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14102: Procedure AABBfromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14103: Function AABBIntersection( const aabb1, aabb2 : TAABB ) : TAABB
14104: Function BBTToAABB( const aBB : THmgBoundingBox ) : TAABB
14105: Function AABBToBB( const anAABB : TAABB ) : THmgBoundingBox
14106: Function AABBToBB1( const anAABB : TAABB; const m : TMatrix ) : THmgBoundingBox
14107: Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14108: Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14109: Function IntersectAABBS( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix ) : Boolean;
14110: Function IntersectAABBS( const aabb1, aabb2 : TAABB ) : Boolean
14111: Function IntersectAABBSAbsoluteXY( const aabb1, aabb2 : TAABB ) : Boolean
14112: Function IntersectAABBSAbsoluteXZ( const aabb1, aabb2 : TAABB ) : Boolean
14113: Function IntersectAABBSAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14114: Function AABBfitsInAABBAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14115: Function PointInAABB( const p : TAffineVector; const aabb : TAABB ) : Boolean;
14116: Function PointInAABB1( const p : TVector; const aabb : TAABB ) : Boolean;
14117: Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB ) : boolean
14118: Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector ) : boolean
14119: Procedure ExtractAABB_corners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14120: Procedure AABBToSphere( const ABB : TAABB; var BSphere : TBSphere)
14121: Procedure BSphereToAABB( const BSphere : TBSphere; var ABB : TAABB );
14122: Function BSphereToAABB1( const center : TAffineVector; radius : Single ) : TAABB;
14123: Function AABBContainsAABB( const mainAABB, testAABB : TAABB ) : TSpaceContains
14124: Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB ) : TSpaceContains
14125: Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere ) : TSpaceContains
14126: Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere ) : TSpaceContains
14127: Function PlaneContainsBSphere( const Location, Normal : TAffineVector; const
14128: testBSphere : TBSphere ) : TSpaceContains
14129: Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere ) : TSpaceContains
14130: Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB ) : TSpaceContains
14131: Function ClipToAABB( const v : TAffineVector; const AABB : TAABB ) : TAffineVector
14132: Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14133: Function AABBToClipRect( const aabb : TAABB; modelViewProjection : TMatrix; viewportSizeX,
14134: viewportSizeY : Int ) : TClipRect
14135: end;

```

```

14136: procedure SIRegister_GeometryCoordinates(CL: TPSPPascalCompiler);
14137: begin
14138:   Procedure Cylindrical_Cartesian( const r, theta, zl : single; var x, y, z : single);
14139:   Procedure Cylindrical_Cartesian1( const r, theta, zl : double; var x, y, z : double);
14140:   Procedure Cylindrical_Cartesian2( const r,theta,zl : single; var x, y, z : single; var ierr : integer);
14141:   Procedure Cylindrical_Cartesian3( const r,theta,zl : double; var x, y, z : double; var ierr : integer);
14142:   Procedure Cartesian_Cylindrical( const x, y, zl : single; var r, theta, z : single);
14143:   Procedure Cartesian_Cylindrical1( const x, y, zl : double; var r, theta, z : double);
14144:   Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14145:   Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14146:   Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14147:   Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14148:   Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14149:   Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14150:   Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14151:   Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14152:   Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14153:   Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14154:   Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14155:   Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14156:   Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14157:   Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr:integer);
14158:   Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14159:   Procedure BipolarCylindrical_Cartesian( const u, v, zl, a : single; var x, y, z : single);
14160:   Procedure BipolarCylindrical_Cartesian1(const u, v, zl, a : double; var x, y, z : double);
14161:   Procedure BipolarCylindrical_Cartesian2(const u,v,zl,a: single;var x,y,z:single; var ierr : integer);
14162:   Procedure BipolarCylindrical_Cartesian3(const u,v,zl,a: double;var x,y,z:double; var ierr : integer);
14163: end;
14164:
14165: procedure SIRegister_VectorGeometry(CL: TPSPPascalCompiler);
14166: begin
14167:   'EPSILON','Single').setExtended( 1e-40);
14168:   'EPSILON2','Single').setExtended( 1e-30); }
14169:   TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14170:   +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14171:   THmgPlane', 'TVector
14172:   TDoubleshmgPlane', 'THomogeneousDblVector
14173:   {TTransType', '({ ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14174:   +'XZ, ttShearyZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14175:   +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14176:   TSingleArray', 'array of Single
14177:   TTransformations', 'array [0..15] of Single)
14178:   TPackedRotationMatrix', 'array [0..2] of Smallint)
14179:   TVertex', 'TAffineVector
14180:   //TVectorGL', 'THomogeneousFltVector
14181:   //TMATRIXGL', 'THomogeneousFltMatrix
14182:   // TPackedRotationMatrix = array [0..2] of SmallInt;
14183:   Function glTexPointMake( const s, t : Single) : TTExPoint
14184:   Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14185:   Function glAffineVectorMakel( const v : TVectorGL) : TAffineVector;
14186:   Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14187:   Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14188:   Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14189:   Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14190:   Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14191:   Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14192:   Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14193:   Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14194:   Function glPointMake( const x, y, z : Single) : TVectorGL;
14195:   Function glPointMakel( const v : TAffineVector) : TVectorGL;
14196:   Function glPointMake2( const v : TVectorGL) : TVectorGL;
14197:   Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14198:   Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14199:   Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14200:   Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14201:   Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14202:   Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14203:   Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14204:   Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14205:   Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14206:   Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14207:   Procedure glRstVector( var v : TAffineVector);
14208:   Procedure glRstVector1( var v : TVectorGL);
14209:   Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14210:   Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14211:   //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14212:   Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14213:   Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14214:   Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14215:   Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14216:   Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14217:   Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14218:   Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14219:   Procedure glAddVector10( var v : TAffineVector; const f : Single);
14220:   Procedure glAddVector11( var v : TVectorGL; const f : Single);
14221:   //Procedure TexPointArrayAdd(const src:PTexPointArray,const delta:TTexPoint,const
14222:   nb:Int;dest:PtexPointArray);
14222:   //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray,const delta:TTexPoint,const
14222:   nb:Integer;const scale: TTExPoint; dest : PtexPointArray);

```

```

14223: //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector,const nb:Integer;dest:
14224: PAffineVectorArray);
14225: Function glVectorSubtract( const V1, V2 : TAffineVector ) : TAffineVector;
14226: Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector );
14227: Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL );
14228: Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL );
14229: Function glVectorSubtract4( const V1, V2 : TVectorGL ) : TVectorGL;
14230: Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL );
14231: Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector );
14232: Function glVectorSubtract7( const v1 : TAffineVector; delta : Single ) : TAffineVector;
14233: Function glVectorSubtract8( const v1 : TVectorGL; delta : Single ) : TVectorGL;
14234: Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector );
14235: Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL );
14236: //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat );
14237: Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single ) : TTexPoint;
14238: Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single ) : TAffineVector;
14239: Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single ) : TAffineVector;
14240: Procedure glVectorCombine34( const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector );
14241: Procedure glCombineVector5( var vr : TVectorGL; const F1, F2 : Single );
14242: Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single );
14243: Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single ) : TVectorGL;
14244: Function glVectorCombine8( const V1 : TVectorGL; const V2 : TAffineVector; const F1,F2:Single );
14245: Procedure glVectorCombine9( const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL );
14246: Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL );
14247: Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL );
14248: Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single ) : TVectorGL;
14249: Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL );
14250: Function glVectorDotProduct( const V1, V2 : TAffineVector ) : Single;
14251: Function glVectorDotProduct1( const V1, V2 : TVectorGL ) : Single;
14252: Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector ) : Single;
14253: Function glPointProject( const p, origin, direction : TAffineVector ) : Single;
14254: Function glPointProject1( const p, origin, direction : TVectorGL ) : Single;
14255: Function glVectorCrossProduct( const V1, V2 : TAffineVector ) : TAffineVector;
14256: Function glVectorCrossProduct1( const V1, V2 : TVectorGL ) : TVectorGL;
14257: Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL );
14258: Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL );
14259: Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector );
14260: Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector );
14261: Function glLerp( const start, stop, t : Single ) : Single;
14262: Function glAngleLerp( start, stop, t : Single ) : Single;
14263: Function glDistanceBetweenAngles( angle1, angle2 : Single ) : Single;
14264: Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single ) : TTexPoint;
14265: Function glVectorLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14266: Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector );
14267: Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single ) : TVectorGL;
14268: Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL );
14269: Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14270: Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single ) : TAffineVector;
14271: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray );
14272: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
14273: PAffineVectorArray );
14274: Function glVectorLength( const x, y : Single ) : Single;
14275: Function glVectorLength1( const x, y, z : Single ) : Single;
14276: Function glVectorLength2( const v : TAffineVector ) : Single;
14277: Function glVectorLength3( const v : TVectorGL ) : Single;
14278: Function glVectorLength4( const v : array of Single ) : Single;
14279: Function glVectorNorm( const x, y : Single ) : Single;
14280: Function glVectorNorm1( const v : TAffineVector ) : Single;
14281: Function glVectorNorm2( const v : TVectorGL ) : Single;
14282: Function glVectorNorm3( var V : array of Single ) : Single;
14283: Procedure glNormalizeVector( var v : TAffineVector );
14284: Procedure glNormalizeVector1( var v : TVectorGL );
14285: Function glVectorNormalize( const v : TAffineVector ) : TAffineVector;
14286: Function glVectorNormalize1( const v : TVectorGL ) : TVectorGL;
14287: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer );
14288: Function glVectorAngleCosine( const V1, V2 : TAffineVector ) : Single;
14289: Function glVectorNegate( const v : TAffineVector ) : TAffineVector;
14290: Function glVectorNegate1( const v : TVectorGL ) : TVectorGL;
14291: Procedure glNegateVector( var V : TAffineVector );
14292: Procedure glNegateVector2( var V : TVectorGL );
14293: Procedure glNegateVector3( var V : array of Single );
14294: Procedure glScaleVector( var v : TAffineVector; factor : Single );
14295: Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector );
14296: Procedure glScaleVector2( var v : TVectorGL; factor : Single );
14297: Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL );
14298: Function glVectorScale( const v : TAffineVector; factor : Single ) : TAffineVector;
14299: Function glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector );
14300: Procedure glVectorScale2( const v : TVectorGL; factor : Single ) : TVectorGL;
14301: Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL );
14302: Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector );
14303: Function glDivideVector( var v : TVectorGL; const divider : TVectorGL );
14304: Function glVectorEquals( const V1, V2 : TVectorGL ) : Boolean;
14305: Function glAffineVectorEquals( const V1, V2 : TVectorGL ) : Boolean;
14306: Function glVectorIsNull( const v : TVectorGL ) : Boolean;
14307: Function glVectorIsNotNull( const v : TAffineVector ) : Boolean;
14308: Function glVectorSpacing( const v1, v2 : TTExPoint ) : Single;
14309: Function glVectorSpacing1( const v1, v2 : TAffineVector ) : Single;

```

```

14310: Function glVectorSpacing2( const v1, v2 : TVectorGL ) : Single;
14311: Function glVectorDistance( const v1, v2 : TAffineVector ) : Single;
14312: Function glVectorDistance1( const v1, v2 : TVectorGL ) : Single;
14313: Function glVectorDistance2( const v1, v2 : TAffineVector ) : Single;
14314: Function glVectorDistance21( const v1, v2 : TVectorGL ) : Single;
14315: Function glVectorPerpendicular( const V, N : TAffineVector ) : TAffineVector;
14316: Function glVectorReflect( const V, N : TAffineVector ) : TAffineVector;
14317: Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single );
14318: Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single );
14319: Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single );
14320: Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single ) : TAffineVector;
14321: Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single ) : TAffineVector;
14322: Procedure glVectorRotateAroundY1( const v : TAffineVector; alpha : Single; var vr : TAffineVector );
14323: Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single ) : TAffineVector;
14324: Procedure glAbsVector( var v : TVectorGL );
14325: Procedure glabsVector1( var v : TAffineVector );
14326: Function glVectorAbs( const v : TVectorGL ) : TVectorGL;
14327: Function glVectorAbs1( const v : TAffineVector ) : TAffineVector;
14328: Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL );
14329: Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL );
14330: Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix );
14331: Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL );
14332: Function glCreateScaleMatrix( const v : TAffineVector ) : TMatrixGL;
14333: Function glCreateScaleMatrix1( const v : TVectorGL ) : TMatrixGL;
14334: Function glCreateTranslationMatrix( const V : TAffineVector ) : TMatrixGL;
14335: Function glCreateTranslationMatrix1( const V : TVectorGL ) : TMatrixGL;
14336: Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL ) : TMatrixGL;
14337: Function glCreateRotationMatrixX( const sine, cosine : Single ) : TMatrixGL;
14338: Function glCreateRotationMatrixX1( const angle : Single ) : TMatrixGL;
14339: Function glCreateRotationMatrixY( const sine, cosine : Single ) : TMatrixGL;
14340: Function glCreateRotationMatrixY1( const angle : Single ) : TMatrixGL;
14341: Function glCreateRotationMatrixZ( const sine, cosine : Single ) : TMatrixGL;
14342: Function glCreateRotationMatrixZ1( const angle : Single ) : TMatrixGL;
14343: Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single ) : TMatrixGL;
14344: Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single ) : TMatrixGL;
14345: Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single ):TAffineMatrix;
14346: Function glMatrixMultiply( const M1, M2 : TAffineMatrix ) : TAffineMatrix;
14347: Function glMatrixMultiply1( const M1, M2 : TMatrixGL ) : TMatrixGL;
14348: Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL );
14349: Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL ) : TVectorGL;
14350: Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix ) : TVectorGL;
14351: Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL ) : TAffineVector;
14352: Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix ) : TAffineVector;
14353: Function glMatrixDeterminant( const M : TAffineMatrix ) : Single;
14354: Function glMatrixDeterminant1( const M : TMatrixGL ) : Single;
14355: Procedure glAdjointMatrix( var M : TMatrixGL );
14356: Procedure glAdjointMatrix1( var M : TAffineMatrix );
14357: Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single );
14358: Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single );
14359: Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector );
14360: Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL );
14361: Procedure glNormalizeMatrix( var M : TMatrixGL );
14362: Procedure glTransposeMatrix( var M : TAffineMatrix );
14363: Procedure glTransposeMatrix1( var M : TMatrixGL );
14364: Procedure glInvertMatrix( var M : TMatrixGL );
14365: Procedure glInvertMatrix1( var M : TAffineMatrix );
14366: Function glAnglePreservingMatrixInvert( const mat : TMatrixGL ) : TMatrixGL;
14367: Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations ) : Boolean;
14368: Function glPlaneMake( const p1, p2, p3 : TAffineVector ) : THmgPlane;
14369: Function glPlaneMake1( const p1, p2, p3 : TVectorGL ) : THmgPlane;
14370: Function glPlaneMake2( const point, normal : TAffineVector ) : THmgPlane;
14371: Function glPlaneMake3( const point, normal : TVectorGL ) : THmgPlane;
14372: Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane );
14373: Procedure glNormalizePlane( var plane : THmgPlane );
14374: Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector ) : Single;
14375: Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL ) : Single;
14376: Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector ) : TAffineVector;
14377: Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector );
14378: Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector );
14379: Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL ) : Boolean;
14380: Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector ) : Boolean;
14381: Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL ) : Single;
14382: Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector ) : Single;
14383: Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector;
14384: Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector ) : single;
14385: Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector ) : TAffineVector;
14386: Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector ) : Single;
14387: Procedure SglementSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var Segment0Closest, Segment1Closest : TAffineVector );
14388: Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single;
14389: TEulerOrder', '( eulXYZ, eulXZY, eulyZX, eulyZX, eulZXY, eulZYX )';
14390: Function glQuaternionMake( const Imag : array of Single; Real : Single ) : TQuaternion;
14391: Function glQuaternionConjugate( const Q : TQuaternion ) : TQuaternion;
14392: Function glQuaternionMagnitude( const Q : TQuaternion ) : Single;
14393: Procedure glNormalizeQuaternion( var Q : TQuaternion );
14394: Function glQuaternionFromPoints( const V1, V2 : TAffineVector ) : TQuaternion;
14395: Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector );
14396: Function glQuaternionFromMatrix( const mat : TMatrixGL ) : TQuaternion;
14397: Function glQuaternionToMatrix( quat : TQuaternion ) : TMatrixGL;

```

```

14398: Function glQuaternionToAffineMatrix( quat : TQuaternion ) : TAffineMatrix
14399: Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector ) : TQuaternion
14400: Function glQuaternionFromRollPitchYaw( const r, p, y : Single ) : TQuaternion
14401: Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder ) : TQuaternion
14402: Function glQuaternionMultiply( const qL, qR : TQuaternion ) : TQuaternion
14403: Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single ) : TQuaternion;
14404: Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single ) : TQuaternion;
14405: Function glLnXp1( X : Extended ) : Extended
14406: Function glLog10( X : Extended ) : Extended
14407: Function glLog2( X : Extended ) : Extended;
14408: Function glLog21( X : Single ) : Single;
14409: Function glLogN( Base, X : Extended ) : Extended
14410: Function glIntPower( Base : Extended; Exponent : Integer ) : Extended
14411: Function glPower( const Base, Exponent : Single ) : Single;
14412: Function glPower1( Base : Single; Exponent : Integer ) : Single;
14413: Function glDegToRad( const Degrees : Extended ) : Extended;
14414: Function glDegToRad1( const Degrees : Single ) : Single;
14415: Function glRadToDeg( const Radians : Extended ) : Extended;
14416: Function glRadToDeg1( const Radians : Single ) : Single;
14417: Function glNormalizeAngle( angle : Single ) : Single
14418: Function glNormalizeDegAngle( angle : Single ) : Single
14419: Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended );
14420: Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double );
14421: Procedure glSinCos( const Theta : Single; var Sin, Cos : Single );
14422: Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended );
14423: Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double );
14424: Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single );
14425: Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single )
14426: Function glArcCos( const X : Extended ) : Extended;
14427: Function glArcCos1( const x : Single ) : Single;
14428: Function glArcSin( const X : Extended ) : Extended;
14429: Function glArcSin1( const X : Single ) : Single;
14430: Function glArcTan2l( const Y, X : Extended ) : Extended;
14431: Function glArcTan21( const Y, X : Single ) : Single;
14432: Function glFastArcTan2( y, x : Single ) : Single
14433: Function glTan( const X : Extended ) : Extended;
14434: Function glTan1( const X : Single ) : Single;
14435: Function glCoTan( const X : Extended ) : Extended;
14436: Function glCoTan1( const X : Single ) : Single;
14437: Function glSinh( const x : Single ) : Single;
14438: Function glSinh1( const x : Double ) : Double;
14439: Function glCosh( const x : Single ) : Single;
14440: Function glCosh1( const x : Double ) : Double;
14441: Function glRSqrt( v : Single ) : Single
14442: Function glRLength( x, y : Single ) : Single
14443: Function glISqrt( i : Integer ) : Integer
14444: Function glILength( x, y : Integer ) : Integer;
14445: Function glILength1( x, y, z : Integer ) : Integer;
14446: Procedure glRegisterBasedExp
14447: Procedure glRandomPointOnSphere( var p : TAffineVector )
14448: Function glRoundInt( v : Single ) : Single;
14449: Function glRoundInt1( v : Extended ) : Extended;
14450: Function glTrunc( v : Single ) : Integer;
14451: Function glTrunc64( v : Extended ) : Int64;
14452: Function glInt( v : Single ) : Single;
14453: Function glInt1( v : Extended ) : Extended;
14454: Function glFrac( v : Single ) : Single;
14455: Function glFract( v : Extended ) : Extended;
14456: Function glRound( v : Single ) : Integer;
14457: Function glRound64( v : Single ) : Int64;
14458: Function glRound641( v : Extended ) : Int64;
14459: Function glTrunc( X : Extended ) : Int64
14460: Function glRound( X : Extended ) : Int64
14461: Function glFrac( X : Extended ) : Extended
14462: Function glCeil( v : Single ) : Integer;
14463: Function glCeil64( v : Extended ) : Int64;
14464: Function glFloor( v : Single ) : Integer;
14465: Function glFloor64( v : Extended ) : Int64;
14466: Function glScaleAndRound( i : Integer; var s : Single ) : Integer
14467: Function glSign( x : Single ) : Integer
14468: Function glIsInRange( const x, a, b : Single ) : Boolean;
14469: Function glIsInRangel( const x, a, b : Double ) : Boolean;
14470: Function glIsInCube( const p, d : TAffineVector ) : Boolean;
14471: Function glIsInCubel( const p, d : TVectorGL ) : Boolean;
14472: //Function MinFloat( values : PSingleArray; nbItems : Integer ) : Single;
14473: //Function MinFloat1( values : PDoubleArray; nbItems : Integer ) : Double;
14474: //Function MinFloat2( values : PExtendedArray; nbItems : Integer ) : Extended;
14475: Function glMinFloat3( const v1, v2 : Single ) : Single;
14476: Function glMinFloat4( const v : array of Single ) : Single;
14477: Function glMinFloat5( const v1, v2 : Double ) : Double;
14478: Function glMinFloat6( const v1, v2 : Extended ) : Extended;
14479: Function glMinFloat7( const v1, v2, v3 : Single ) : Single;
14480: Function glMinFloat8( const v1, v2, v3 : Double ) : Double;
14481: Function glMinFloat9( const v1, v2, v3 : Extended ) : Extended;
14482: //Function MaxFloat10( values : PSingleArray; nbItems : Integer ) : Single;
14483: //Function MaxFloat( values : PDoubleArray; nbItems : Integer ) : Double;
14484: //Function MaxFloat1( values : PExtendedArray; nbItems : Integer ) : Extended;
14485: Function glMaxFloat2( const v : array of Single ) : Single;
14486: Function glMaxFloat3( const v1, v2 : Single ) : Single;

```

```

14487: Function glMaxFloat4( const v1, v2 : Double ) : Double;
14488: Function glMaxFloat5( const v1, v2 : Extended ) : Extended;
14489: Function glMaxFloat6( const v1, v2, v3 : Single ) : Single;
14490: Function glMaxFloat7( const v1, v2, v3 : Double ) : Double;
14491: Function glMaxFloat8( const v1, v2, v3 : Extended ) : Extended;
14492: Function glMinInteger9( const v1, v2 : Integer ) : Integer;
14493: Function glMinInteger( const v1, v2 : Cardinal ) : Cardinal;
14494: Function glMaxInteger( const v1, v2 : Integer ) : Integer;
14495: Function glMaxInteger1( const v1, v2 : Cardinal ) : Cardinal;
14496: Function glTriangleArea( const p1, p2, p3 : TAffineVector ) : Single;
14497: //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer ) : Single;
14498: Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector ) : Single;
14499: //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer ) : Single;
14500: //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single );
14501: Procedure glScaleFloatArray( var values : TSingleArray; factor : Single );
14502: //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single );
14503: Procedure glOffsetFloatArray( var values : array of Single; delta : Single );
14504: //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer );
14505: Function glMaxXYZComponent( const v : TVectorGL ) : Single;
14506: Function glMaxXYZComponent1( const v : TAffineVector ) : single;
14507: Function glMinXYZComponent( const v : TVectorGL ) : Single;
14508: Function glMinXYZComponent1( const v : TAffineVector ) : single;
14509: Function glMaxAbsXYZComponent( v : TVectorGL ) : Single;
14510: Function glMinAbsXYZComponent( v : TVectorGL ) : Single;
14511: Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL );
14512: Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector );
14513: Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL );
14514: Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector );
14515: Procedure glSortArrayAscending( var a : array of Extended );
14516: Function glClampValue( const aValue, aMin, aMax : Single ) : Single;
14517: Function glClampValue1( const aValue, aMin : Single ) : Single;
14518: Function glGeometryOptimizationMode : String;
14519: Procedure glBeginFPUOnlySection;
14520: Procedure glEndFPUOnlySection;
14521: Function glConvertRotation( const Angles : TAffineVector ) : TVectorGL;
14522: Function glMakeAffineDblVector( var v : array of Double ) : TAffineDblVector;
14523: Function glMakeDblVector( var v : array of Double ) : THomogeneousDblVector;
14524: Function glVectorAffineDblToFlt( const v : TAffineDblVector ) : TAffineVector;
14525: Function glVectorDblToFlt( const v : THomogeneousDblVector ) : THomogeneousVector;
14526: Function glVectorAffineFltToDbl( const v : TAffineVector ) : TAffineDblVector;
14527: Function glVectorFltToDbl( const v : TVectorGL ) : THomogeneousDblVector;
14528: Function glPointInPolygon( var xp, yp : array of Single; x, y : Single ) : Boolean;
14529: Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word );
14530: Function glTurn( const Matrix : TMatrixGL; angle : Single ) : TMatrixGL;
14531: Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14532: Function glPitch( const Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
14533: Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14534: Function glRoll( const Matrix: TMatrixGL; Angle : Single ) : TMatrixGL;
14535: Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14536: Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single;
14537: Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;const sphereRadius : Single ) : Boolean;
14538: Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL; const sphereRadius : Single; var i1, i2 : TVectorGL ) : Integer;
14539: Function glSphereVisibleRadius( distance, radius : Single ) : Single;
14540: Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL ) : TFrustum;
14541: Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci : TRenderContextClippingInfo ) : Boolean;
14542: Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci : TRenderContextClippingInfo ) : Boolean;
14543: Function glIsVolumeClipped2( const min,max:TAffineVector;const rcci : TRenderContextClippingInfo ) : Bool;
14544: Function glIsVolumeClipped3( const objPos:TAffineVector;const objRadius:Single;const Frustum:TFrustum ):Bool;
14545: Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL ) : TMatrixGL;
14546: Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL ) : TMatrixGL;
14547: Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector ) : TMatrixGL;
14548: Function glPackRotationMatrix( const mat : TMatrixGL ) : TPackedRotationMatrix;
14549: Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix ) : TMatrixGL;
14550: 'cPI','Single').setExtended( 3.141592654 );
14551: 'cPIdiv180','Single').setExtended( 0.017453292 );
14552: 'c180divPI','Single').setExtended( 57.29577951 );
14553: 'c2PI','Single').setExtended( 6.283185307 );
14554: 'cPIdiv2','Single').setExtended( 1.570796326 );
14555: 'cPIdiv4','Single').setExtended( 0.785398163 );
14556: 'c3PIdiv4','Single').setExtended( 2.35619449 );
14557: 'cInv2PI','Single').setExtended( 1 / 6.283185307 );
14558: 'cInv360','Single').setExtended( 1 / 360 );
14559: 'c180','Single').setExtended( 180 );
14560: 'c360','Single').setExtended( 360 );
14561: 'cOneHalf','Single').setExtended( 0.5 );
14562: 'cLn10','Single').setExtended( 2.302585093 );
14563: {'MinSingle','Extended').setExtended( 1.5e-45 );
14564: 'MaxSingle','Extended').setExtended( 3.4e+38 );
14565: 'MinDouble','Extended').setExtended( 5.0e-324 );
14566: 'MaxDouble','Extended').setExtended( 1.7e+308 );
14567: 'MinExtended','Extended').setExtended( 3.4e-4932 );
14568: 'MaxExtended','Extended').setExtended( 1.1e+4932 );
14569: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18 );
14570: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18 );}

```

```

14571: end;
14572:
14573: procedure SIRegister_GLVectorFileObjects(CL: TPSPPascalCompiler);
14574: begin
14575:   AddClassN(FindClass('TOBJECT'), 'TMeshObjectList'
14576:   (FindClass('TOBJECT'), 'TFaceGroups'
14577:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14578:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14579:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14580:   SIRegister_TBaseMeshObject(CL);
14581:   (FindClass('TOBJECT'), 'TSkeletonFrameList'
14582:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14583:   SIRegister_TSkeletonFrame(CL);
14584:   SIRegister_TSkeletonFrameList(CL);
14585:   (FindClass('TOBJECT'), 'TSkeleton
14586:   (FindClass('TOBJECT'), 'TSkeletonBone
14587:   SIRegister_TSkeletonBoneList(CL);
14588:   SIRegister_TSkeletonRootBoneList(CL);
14589:   SIRegister_TSkeletonBone(CL);
14590:   (FindClass('TOBJECT'), 'TSkeletonColliderList
14591:   SIRegister_TSkeletonCollider(CL);
14592:   SIRegister_TSkeletonColliderList(CL);
14593:   (FindClass('TOBJECT'), 'TGLBaseMesh
14594:   TMeshLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14595:   +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14596:   +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14597:   +'QuaternionList; end
14598:   SIRegister_TSkeleton(CL);
14599:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14600:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14601:   SIRegister_TMeshObject(CL);
14602:   SIRegister_TMeshObjectList(CL);
14603: //TMeshObjectListClass', 'class of TMeshObjectList
14604:   (FindClass('TOBJECT'), 'TMeshMorphTargetList
14605:   SIRegister_TMeshMorphTarget(CL);
14606:   SIRegister_TMeshMorphTargetList(CL);
14607:   SIRegister_TMorphableMeshObject(CL);
14608:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14609: //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not work
14610: //FVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14611: TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14612:   SIRegister_TSkeletonMeshObject(CL);
14613:   SIRegister_TFaceGroup(CL);
14614:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14615:   +'atTriangles, fgmmTriangleFan, fgmmQuads )
14616:   SIRegister_TFGVertexIndexList(CL);
14617:   SIRegister_TFGVertexNormalTexIndexList(CL);
14618:   SIRegister_TFGIndexTexCoordList(CL);
14619:   SIRegister_TFaceGroups(CL);
14620:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14621:   SIRegister_TVectorFile(CL);
14622: //TVectorFileClass', 'class of TVectorFile
14623:   SIRegister_TGLGLSMVectorFile(CL);
14624:   SIRegister_TGLBaseMesh(CL);
14625:   SIRegister_TGLFreeForm(CL);
14626:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14627:   TGLActorOptions', 'set of TGLActorOption
14628: 'cDefaultGLActorOptions', 'LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14629:   (FindClass('TOBJECT'), 'TGLActor
14630:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14631:   SIRegister_TActorAnimation(CL);
14632:   TActorAnimationName', 'String
14633:   SIRegister_TActorAnimations(CL);
14634:   SIRegister_TGLBaseAnimationController(CL);
14635:   SIRegister_TGLAnimationController(CL);
14636:   TActorFrameInterpolation', '( afpNone, afpLinear )
14637:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounce'
14638:   +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14639:   SIRegister_TGLActor(CL);
14640:   SIRegister_TVectorFileFormat(CL);
14641:   SIRegister_TVectorFileFormatsList(CL);
14642:   (FindClass('TOBJECT'), 'EInvalidVectorFile
14643:   Function GetVectorFileFormats : TVectorFileFormatsList
14644:   Function VectorFileFormatsFilter : String
14645:   Function VectorFileFormatsSaveFilter : String
14646:   Function VectorFileFormatExtensionByIndex( index : Integer ) : String
14647:   Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass )
14648:   Procedure UnregisterVectorFileClass( aClass : TVectorFileClass )
14649: end;
14650:
14651: procedure SIRegister_AxCtrls(CL: TPSPPascalCompiler);
14652: begin
14653:   'Class_DColorPropPage', 'TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14654:   'Class_DFontPropPage', 'TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14655:   'Class_DPicturePropPage', 'TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14656:   'Class_DStringPropPage', 'TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14657:   SIRegister_ToleStream(CL);
14658:   (FindClass('TOBJECT'), 'TConnectionPoints
14659:   TConnectionKind', '( ckSingle, ckMulti )

```

```

14660: SIRегистер_TConnectionPoint(CL);
14661: SIRегистер_TConnectionPoints(CL);
14662: TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14663: (FindClass('TOBJECT'), 'TActiveXControlFactory
14664: SIRегистер_TActiveXControl(CL);
14665: //TActiveXControlClass', 'class of TActiveXControl
14666: SIRегистер_TActiveXControlFactory(CL);
14667: SIRегистер_TActiveFormControl(CL);
14668: SIRегистер_TActiveForm(CL);
14669: //TActiveFormClass', 'class of TActiveForm
14670: SIRегистер_TActiveFormFactory(CL);
14671: (FindClass('TOBJECT'), 'TPropertyPageImpl
14672: SIRегистер_TPropertyPage(CL);
14673: //TPropertyPageClass', 'class of TPropertyPage
14674: SIRегистер_TPropertyPageImpl(CL);
14675: SIRегистер_TActiveXPropertyPage(CL);
14676: SIRегистер_TActiveXPropertyPageFactory(CL);
14677: SIRегистер_TCustomAdapter(CL);
14678: SIRегистер_TAdapterNotifier(CL);
14679: SIRегистер_IFontAccess(CL);
14680: SIRегистер_TFontAdapter(CL);
14681: SIRегистер_IPictureAccess(CL);
14682: SIRегистер_TPictureAdapter(CL);
14683: SIRегистер_TOLEGraphic(CL);
14684: SIRегистер_TStringsAdapter(CL);
14685: SIRегистер_TReflectorWindow(CL);
14686: Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:GUID;VTCode:Int;PropList:TStrings);
14687: Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14688: Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14689: Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14690: Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14691: Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14692: Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14693: Function ParkingWindow : HWND
14694: end;
14695:
14696: procedure SIRегистер_synaip(CL: TPSPPascalCompiler);
14697: begin
14698: // TIP6Bytes = array [0..15] of Byte;
14699: {:binary form of IPv6 adress (for string conversion routines)}
14700: // TIP6Words = array [0..7] of Word;
14701: AddTypeS('TIP6Bytes', 'array [0..15] of Byte;');
14702: AddTypes('TIP6Words', 'array [0..7] of Word;');
14703: AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14704: Function synaIsIP( const Value : string) : Boolean';
14705: Function synaIsIP6( const Value : string) : Boolean';
14706: Function synaPToID( Host : string) : Ansistring';
14707: Function synaStrToIp6( value : string) : TIP6Bytes';
14708: Function synaIp6ToStr( value : TIP6Bytes) : string';
14709: Function synaStrToIp( value : string) : integer';
14710: Function synaIpToStr( value : integer) : string';
14711: Function synaReverseIP( Value : AnsiString) : AnsiString';
14712: Function synaReverseIP6( Value : AnsiString) : AnsiString';
14713: Function synaExpandIP6( Value : AnsiString) : AnsiString';
14714: Function xStrToIP( const Value : String) : TIPAdr';
14715: Function xIPToStr( const Adresse : TIPAdr) : String';
14716: Function IPToCardinal( const Adresse : TIPAdr) : Cardinal';
14717: Function CardinalToIP( const Value : Cardinal) : TIPAdr';
14718: end;
14719:
14720: procedure SIRегистер_synacode(CL: TPSPPascalCompiler);
14721: begin
14722: AddTypeS('TSpecials', 'set of Char');
14723: Const('SpecialChar', 'TSpecials').SetSet( '=( )[]<>;,@/?\"_');
14724: Const('URLFullSpecialChar', 'TSpecials').SetSet( ';/?:@=&#'+');
14725: Const('TableBase64' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghi jklmnopqrstuvwxyz0123456789+/=+');
14726: Const('TableBase64mod' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghi jklmnopqrstuvwxyz0123456789+,=+');
14727: Const('TableUU' '!#$%&()' *+-./0123456789;:<=>@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14728: Const('TableXX' (+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghi jklmnopqrstuvwxyz');
14729: Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString';
14730: Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString';
14731: Function DecodeURL( const Value : AnsiString) : AnsiString';
14732: Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString';
14733: Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString';
14734: Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString';
14735: Function EncodeURLElement( const Value : AnsiString) : AnsiString';
14736: Function EncodeURL( const Value : AnsiString) : AnsiString';
14737: Function Decode4to3( const Value, Table : AnsiString) : AnsiString';
14738: Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString';
14739: Function Encode3to4( const Value, Table : AnsiString) : AnsiString';
14740: Function synDecodeBase64( const Value : AnsiString) : AnsiString';
14741: Function synEncodeBase64( const Value : AnsiString) : AnsiString';
14742: Function DecodeBase64mod( const Value : AnsiString) : AnsiString';
14743: Function EncodeBase64mod( const Value : AnsiString) : AnsiString';
14744: Function DecodeUU( const Value : AnsiString) : AnsiString';
14745: Function EncodeUU( const Value : AnsiString) : AnsiString';
14746: Function DecodeXX( const Value : AnsiString) : AnsiString';
14747: Function DecodeYEnc( const Value : AnsiString) : AnsiString';
14748: Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer';

```

```

14749: Function synCrc32( const Value : AnsiString ) : Integer';
14750: Function UpdateCrc16( Value : Byte; Crc16 : Word ) : Word';
14751: Function Crc16( const Value : AnsiString ) : Word';
14752: Function synMD5( const Value : AnsiString ) : AnsiString';
14753: Function HMAC_MD5( Text, Key : AnsiString ) : AnsiString';
14754: Function MD5LongHash( const Value : AnsiString; Len : integer ) : AnsiString';
14755: Function synSHA1( const Value : AnsiString ) : AnsiString';
14756: Function HMAC_SHA1( Text, Key : AnsiString ) : AnsiString';
14757: Function SHA1LongHash( const Value : AnsiString; Len : integer ) : AnsiString';
14758: Function synMD4( const Value : AnsiString ) : AnsiString';
14759: end;
14760:
14761: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14762: begin
14763:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14764:             +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14765:             +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14766:             +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14767:             +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE, '
14768:             +'C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14769:             +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14770:             +', NEXTSTEP, ARMSCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14771:             +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14772:             +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14773:             +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14774:             +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14775:             +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14776:             +', CP864, CP865, CP869, CP1125 ') );
14777:   AddTypes('TMimeSetChar', 'set of TMimeChar');
14778:   Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14779:   Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
14780:     TransformTable : array of Word) : AnsiString';
14781:   Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
14782:     TransformTable : array of Word; Translit : Boolean) : AnsiString');
14783:   Function GetCurCP : TMimeChar');
14784:   Function GetCuroEMCP : TMimeChar');
14785:   Function GetCPFromID( Value : AnsiString ) : TMimeChar';
14786:   Function GetIDFromCP( Value : TMimeChar ) : AnsiString';
14787:   Function NeedCharsetConversion( const Value : AnsiString ) : Boolean';
14788:   Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14789:   Function GetBOM( Value : TMimeChar ) : AnsiString';
14790:   Function StringToWide( const Value : AnsiString ) : WideString';
14791:   Function WideToString( const Value : WideString ) : AnsiString');
14792: end;
14793: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14794: begin
14795:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14796:   Procedure WakeOnLan( MAC, IP : string );
14797:   Function GetDNS : string';
14798:   Function GetIEProxy( protocol : string ) : TProxySetting');
14799: end;
14800:
14801:
14802: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14803: begin
14804:   AddConstantN('synCR', 'Char #$0d);
14805:   Const('synLF', 'Char #$0a);
14806:   Const('cSerialChunk', 'LongInt'( 8192);
14807:   Const('LockfileDirectory', 'String '/var/lock');
14808:   Const('PortIsClosed', 'LongInt'( - 1);
14809:   Const('ErrAlreadyOwned', 'LongInt'( 9991);
14810:   Const('ErrAlreadyInUse', 'LongInt'( 9992);
14811:   Const('ErrWrongParameter', 'LongInt'( 9993);
14812:   Const('ErrPortNotOpen', 'LongInt'( 9994);
14813:   Const('ErrNoDeviceAnswer', 'LongInt'( 9995);
14814:   Const('ErrMaxBuffer', 'LongInt'( 9996);
14815:   Const('ErrTimeout', 'LongInt'( 9997);
14816:   Const('ErrNotRead', 'LongInt'( 9998);
14817:   Const('ErrFrame', 'LongInt'( 9999);
14818:   Const('ErrOverrun', 'LongInt'( 10000);
14819:   Const('ErrRxOver', 'LongInt'( 10001);
14820:   Const('ErrRxParity', 'LongInt'( 10002);
14821:   Const('ErrTxFull', 'LongInt'( 10003);
14822:   Const('dcb_Binary', 'LongWord')( $00000001);
14823:   Const('dcb_ParityCheck', 'LongWord')( $00000002);
14824:   Const('dcb_OutxCtsFlow', 'LongWord')( $00000004);
14825:   Const('dcb_OutxDsrFlow', 'LongWord')( $00000008);
14826:   Const('dcb_DtrControlMask', 'LongWord')( $00000030);
14827:   Const('dcb_DtrControlDisable', 'LongWord')( $00000000);
14828:   Const('dcb_DtrControlEnable', 'LongWord')( $00000010);
14829:   Const('dcb_DtrControlHandshake', 'LongWord')( $00000020);
14830:   Const('dcb_DsrSensitivity', 'LongWord')( $00000040);
14831:   Const('dcb_TXContinueOnXoff', 'LongWord')( $00000080);
14832:   Const('dcb_OutX', 'LongWord')( $00000100);
14833:   Const('dcb_InX', 'LongWord')( $00000200);
14834:   Const('dcb_ErrorChar', 'LongWord')( $00000400);
14835:   Const('dcb_NullStrip', 'LongWord')( $00000800);

```

```

14836: Const('dcb_RtsControlMask','LongWord')( $00003000);
14837: Const('dcb_RtsControlDisable','LongWord')( $00000000);
14838: Const('dcb_RtsControlEnable','LongWord')( $00001000);
14839: Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14840: Const('dcb_RtsControlToggle','LongWord')( $00003000);
14841: Const('dcb_AbortOnError','LongWord')( $00004000);
14842: Const('dcb_Reserves','LongWord')( $FFFF8000);
14843: Const('synSBl','LongInt'( 0));
14844: Const('SBlandHalf','LongInt'( 1));
14845: Const('synSB2','LongInt'( 2));
14846: Const('synINVALID_HANDLE_VALUE','LongInt'( THandle( - 1)));
14847: Const('CS7fix','LongWord')( $00000020);
14848: AddTypeS('synTDCB','record DCBlength : DWORD; BaudRate : WORD; Flags : Long'
14849:   + 'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14850:   + 'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14851:   + 'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14852: //AddTypeS('PDCB','^TDCB // will not work');
14853: //Const('MaxRates','LongInt'( 18));
14854: //Const('MaxRates','LongInt'( 30));
14855: //Const('MaxRates','LongInt'( 19));
14856: Const('O_SYNC','LongWord')( $0080);
14857: Const('synOK','LongInt'( 0));
14858: Const('synErr','LongInt'( integer( - 1)));
14859: AddTypeS('THookSerialReason','( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
14860:   HR_WriteCount, HR_Wait )');
14861: Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string));
14862: SIRegister_ESynaSerError(CL);
14863: SIRegister_TBlockSerial(CL);
14864: Function GetSerialPortNames : string;
14865: end;
14866: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14867: begin
14868:   Const('DLLIconvName','String 'libiconv.so');
14869:   Const('DLLIconvName','String 'iconv.dll');
14870:   AddTypeS('size_t','Cardinal');
14871:   AddTypeS('iconv_t','Integer');
14872:   //AddTypeS('iconv_t','Pointer');
14873:   AddTypeS('argptr','iconv_t');
14874:   Function SynalconvOpen( const tocode, fromcode : Ansistring ) : iconv_t';
14875:   Function SynalconvOpenSlit( const tocode, fromcode : Ansistring ) : iconv_t';
14876:   Function SynalconvOpenIgnore( const tocode, fromcode : Ansistring ) : iconv_t';
14877:   Function Synalconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString ) : integer';
14878:   Function SynalconvClose( var cd : iconv_t ) : integer';
14879:   Function SynalconvCtl( cd : iconv_t; request : integer; argument : argptr ) : integer';
14880:   Function IsIconvloaded : Boolean';
14881:   Function InitIconvInterface : Boolean';
14882:   Function DestroyIconvInterface : Boolean';
14883:   Const('ICONV_TRIVIALP','LongInt'( 0));
14884:   Const('ICONV_GET_TRANSLITERATE','LongInt'( 1));
14885:   Const('ICONV_SET_TRANSLITERATE','LongInt'( 2));
14886:   Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3));
14887:   Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4));
14888: end;
14889:
14890: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14891: begin
14892:   Const('ICMP_ECHO','LongInt'( 8));
14893:   Const('ICMP_ECHOREPLY','LongInt'( 0));
14894:   Const('ICMP_UNREACH','LongInt'( 3));
14895:   Const('ICMP_TIME_EXCEEDED','LongInt'( 11));
14896:   Const('ICMP6_ECHO','LongInt'( 128));
14897:   Const('ICMP6_ECHOREPLY','LongInt'( 129));
14898:   Const('ICMP6_UNREACH','LongInt'( 1));
14899:   Const('ICMP6_TIME_EXCEEDED','LongInt'( 3));
14900:   AddTypeS('TICMPError','( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOut'
14901:     + 'her, IE_UreachRoute, IE_UreachAdmin, IE_UreachAddr, IE_UreachPort )');
14902:   SIRegister_TPINGSend(CL);
14903:   Function PingHost( const Host : string ) : Integer';
14904:   Function TraceRouteHost( const Host : string ) : string';
14905: end;
14906:
14907: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14908: begin
14909:   AddConstantN('synASN1_BOOL','LongWord')( $01);
14910:   Const('synASN1_INT','LongWord')( $02);
14911:   Const('synASN1_OCTSTR','LongWord')( $04);
14912:   Const('synASN1_NULL','LongWord')( $05);
14913:   Const('synASN1_OBJID','LongWord')( $06);
14914:   Const('synASN1_ENUM','LongWord')( $0a);
14915:   Const('synASN1_SEQ','LongWord')( $30);
14916:   Const('synASN1_SETOF','LongWord')( $31);
14917:   Const('synASN1_IPADDR','LongWord')( $40);
14918:   Const('synASN1_COUNTER','LongWord')( $41);
14919:   Const('synASN1_GAUGE','LongWord')( $42);
14920:   Const('synASN1_TIMETICKS','LongWord')( $43);
14921:   Const('synASN1_OPAQUE','LongWord')( $44);
14922:   Function synASNEncOIDItem( Value : Integer ) : AnsiString';
14923:   Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString ) : Integer';

```

```

14924: Function synASNEncLen( Len : Integer ) : AnsiString';
14925: Function synASNDecLen( var Start : Integer; const Buffer : AnsiString ) : Integer';
14926: Function synASNEncInt( Value : Integer ) : AnsiString';
14927: Function synASNEncUInt( Value : Integer ) : AnsiString';
14928: Function synASNObject( const Data : AnsiString; ASNType : Integer ) : AnsiString';
14929: Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14930: Function synMibToId( Mib : String ) : AnsiString';
14931: Function synIdToMib( const Id : AnsiString ) : String';
14932: Function synIntMibToStr( const Value : AnsiString ) : AnsiString';
14933: Function ASNdump( const Value : AnsiString ) : AnsiString';
14934: Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings ) : Boolean';
14935: Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString';
14936: end;
14937:
14938: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14939: begin
14940:   Const ('cLDAPProtocol','String '389');
14941:   Const ('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14942:   Const ('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14943:   Const ('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14944:   Const ('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14945:   Const ('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14946:   Const ('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14947:   Const ('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14948:   Const ('LDAP_ASN1 MODIFY REQUEST','LongWord')( $66);
14949:   Const ('LDAP_ASN1 MODIFY RESPONSE','LongWord')( $67);
14950:   Const ('LDAP_ASN1_ADD REQUEST','LongWord')( $68);
14951:   Const ('LDAP_ASN1_ADD RESPONSE','LongWord')( $69);
14952:   Const ('LDAP_ASN1_DEL REQUEST','LongWord')( $4A);
14953:   Const ('LDAP_ASN1_DEL RESPONSE','LongWord')( $6B);
14954:   Const ('LDAP_ASN1 MODIFYDN REQUEST','LongWord')( $6C);
14955:   Const ('LDAP_ASN1 MODIFYDN RESPONSE','LongWord')( $6D);
14956:   Const ('LDAP_ASN1_COMPARE REQUEST','LongWord')( $6E);
14957:   Const ('LDAP_ASN1_COMPARE RESPONSE','LongWord')( $6F);
14958:   Const ('LDAP_ASN1_ABANDON REQUEST','LongWord')( $70);
14959:   Const ('LDAP_ASN1 EXT REQUEST','LongWord')( $77);
14960:   Const ('LDAP_ASN1 EXT RESPONSE','LongWord')( $78);
14961:   SIRegister_TLDAPAttribute(CL);
14962:   SIRegister_TLDAPAttributeList(CL);
14963:   SIRegister_TLDAPResult(CL);
14964:   SIRegister_TLDAPResultList(CL);
14965:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14966:   AddTypes('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14967:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14968:   SIRegister_TLDAPSnd(CL);
14969:   Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString';
14970: end;
14971:
14972:
14973: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14974: begin
14975:   Const ('cSysLogProtocol','String '514');
14976:   Const ('FCL_Kernel','LongInt'( 0 );
14977:   Const ('FCL_UserLevel','LongInt'( 1 );
14978:   Const ('FCL_MailSystem','LongInt'( 2 );
14979:   Const ('FCL_System','LongInt'( 3 );
14980:   Const ('FCL_Security','LongInt'( 4 );
14981:   Const ('FCL_Syslogd','LongInt'( 5 );
14982:   Const ('FCL_Printer','LongInt'( 6 );
14983:   Const ('FCL_News','LongInt'( 7 );
14984:   Const ('FCL_UUCP','LongInt'( 8 );
14985:   Const ('FCL_Clock','LongInt'( 9 );
14986:   Const ('FCL_Authorization','LongInt'( 10 );
14987:   Const ('FCL_FTP','LongInt'( 11 );
14988:   Const ('FCL_NTP','LongInt'( 12 );
14989:   Const ('FCL_LogAudit','LongInt'( 13 );
14990:   Const ('FCL_LogAlert','LongInt'( 14 );
14991:   Const ('FCL_Time','LongInt'( 15 );
14992:   Const ('FCL_Local0','LongInt'( 16 );
14993:   Const ('FCL_Local1','LongInt'( 17 );
14994:   Const ('FCL_Local2','LongInt'( 18 );
14995:   Const ('FCL_Local3','LongInt'( 19 );
14996:   Const ('FCL_Local4','LongInt'( 20 );
14997:   Const ('FCL_Local5','LongInt'( 21 );
14998:   Const ('FCL_Local6','LongInt'( 22 );
14999:   Const ('FCL_Local7','LongInt'( 23 );
15000:   Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
15001:   SIRegister_TSyslogMessage(CL);
15002:   SIRegister_TSyslogSend(CL);
15003:   Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const Content:string):Boolean;
15004: end;
15005:
15006:
15007: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
15008: begin
15009:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
15010:   SIRegister_TMessHeader(CL);
15011:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');

```

```

15012:  SIRegister_TMimeMess(CL);
15013: end;
15014:
15015: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
15016: begin
15017:   (FindClass('TOBJECT'), 'TMimePart');
15018:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
15019:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
15020:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
15021:   SIRegister_TMimePart(CL);
15022:   Const('MaxMimeType', 'LongInt'( 25));
15023:   Function GenerateBoundary : string');
15024: end;
15025:
15026: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
15027: begin
15028:   Function InlineDecode( const Value : string; CP : TMimeChar) : string');
15029:   Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
15030:   Function NeedInline( const Value : AnsiString) : boolean');
15031:   Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
15032:   Function InlineCode( const Value : string) : string');
15033:   Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
15034:   Function InlineEmail( const Value : string) : string');
15035: end;
15036:
15037: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
15038: begin
15039:   Const('cFtpProtocol', 'String '21');
15040:   Const('cFtpDataProtocol', 'String '20');
15041:   Const('FTP_OK', 'LongInt'( 255);
15042:   Const('FTP_ERR', 'LongInt'( 254);
15043:   AddTypeS('FTTFSStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15044:   SIRegister_TFTPLListRec(CL);
15045:   SIRegister_TFTPLList(CL);
15046:   SIRegister_TFTPSend(CL);
15047:   Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean';
15048:   Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean';
15049:   Function FtpInterServerTransfer(const FromIP,FromPort,FromFile,FromUser,FromPass : string; const ToIP,
ToPort,ToFile,ToUser,ToPass : string) : Boolean';
15050: end;
15051:
15052: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
15053: begin
15054:   Const('cHttpProtocol', 'String '80');
15055:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15056:   SIRegister_THTTPSend(CL);
15057:   Function HttpGetText( const URL : string; const Response : TStrings) : Boolean';
15058:   Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean';
15059:   Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean';
15060:   Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean';
15061:   Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
ResultData:TStrings):Bool;
15062: end;
15063:
15064: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15065: begin
15066:   Const('cSmtpProtocol', 'String '25');
15067:   SIRegister_TSMTSPSend(CL);
15068:   Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
Passw:string):Bool;
15069:   Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15070:   Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
Usrname, Password : string):Boolean';
15071: end;
15072:
15073: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15074: begin
15075:   Const('cSnmpProtocol', 'String '161');
15076:   Const('cSnmpTrapProtocol', 'String '162');
15077:   Const('SNMP_V1', 'LongInt'( 0);
15078:   Const('SNMP_V2C', 'LongInt'( 1);
15079:   Const('SNMP_V3', 'LongInt'( 3);
15080:   Const('PDUGetRequest', 'LongWord')($A0);
15081:   Const('PDUGetNextRequest', 'LongWord')($A1);
15082:   Const('PDUGetResponse', 'LongWord')($A2);
15083:   Const('PDUSetRequest', 'LongWord')($A3);
15084:   Const('PDUTrap', 'LongWord')($A4);
15085:   Const('PDUGetBulkRequest', 'LongWord')($A5);
15086:   Const('PDUInformRequest', 'LongWord')($A6);
15087:   Const('PDUTrapV2', 'LongWord')($A7);
15088:   Const('PDUReport', 'LongWord')($A8);
15089:   Const('ENoError', 'LongInt' 0;
15090:   Const('ETooBig', 'LongInt')($A1);
15091:   Const('ENoSuchName', 'LongInt')($A2);
15092:   Const('EBadValue', 'LongInt')($A3);
15093:   Const('EReadOnly', 'LongInt')($A4);
15094:   Const('EGenErr', 'LongInt')($A5);
15095:   Const('ENoAccess', 'LongInt')($A6);
15096:   Const('EWrongType', 'LongInt')($A7);

```

```

15097: Const('EWrongLength','LongInt'( 8);
15098: Const('EWrongEncoding','LongInt'( 9);
15099: Const('EWrongValue','LongInt'( 10);
15100: Const('ENoCreation','LongInt'( 11);
15101: Const('EInconsistentValue','LongInt'( 12);
15102: Const('EResourceUnavailable','LongInt'( 13);
15103: Const('ECommitFailed','LongInt'( 14);
15104: Const('EUndoFailed','LongInt'( 15);
15105: Const('EAuthorizationError','LongInt'( 16);
15106: Const('ENotWritable','LongInt'( 17);
15107: Const('EInconsistentName','LongInt'( 18);
15108: AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15109: AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15110: AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15111: SIRegister_TSNMPMib(CL);
15112: AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer;
15113:   +EngineTime : integer; EngineStamp : Cardinal; end');
15114: SIRegister_TSNMPRec(CL);
15115: SIRegister_TSNMPSend(CL);
15116: Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean';
15117: Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean';
15118: Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15119: Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15120: Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15121: Function SendTrap(const Dest,Source,Enterprise,Community : AnsiString; Generic, Specific, Seconds : Integer;
15122:   const MIBName, MIBValue : AnsiString; MIBtype : Integer);
15123: Function RecvTrap(var Dest,Source,Enterprise,Community : AnsiString; var Generic, Specific, Seconds : Integer;
15124:   const MIBName, MIBValue : TStringList) : Integer);
15125: end;
15126: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15127: begin
15128:   Function GetDomainName2: AnsiString';
15129:   Function GetDomainController( Domain : AnsiString) : AnsiString';
15130:   Function GetDomainUsers( Controller : AnsiString) : AnsiString';
15131:   Function GetDomainGroups( Controller : AnsiString) : AnsiString';
15132:   Function GetDateTime( Controller : AnsiString) : TDateTime';
15133:   Function GetFullName2( Controller, UserID : AnsiString) : AnsiString';
15134: end;
15135: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15136: begin
15137:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15138:   TwwDateTimeSelection', '(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM)';
15139:   Function wwStrToDate( const S : string) : boolean';
15140:   Function wwStrToTime( const S : string) : boolean';
15141:   Function wwStrToDateTime( const S : string) : boolean';
15142:   Function wwStrToTimeVal( const S : string) : TDateTime';
15143:   Function wwStrToDateVal( const S : string) : TDateTime';
15144:   Function wwStrToDateTimeVal( const S : string) : TDateTime';
15145:   Function wwStrToInt( const S : string) : boolean';
15146:   Function wwStrToFloat( const S : string) : boolean';
15147:   Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder';
15148:   Function wwNextDay( Year, Month, Day : Word) : integer';
15149:   Function wwPriorDay( Year, Month, Day : Word) : integer';
15150:   Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean';
15151:   Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean';
15152:   Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection';
15153:   Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection';
15154:   Function wwScanDate( const S : string; var Date : TDateTime) : Boolean';
15155:   Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean';
15156:   Procedure wwSetDateTimeCursorPosition(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool);
15157:   Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean';
15158: end;
15159:
15160: unit uPSI_Themes;
15161: Function ThemeServices : TThemeServices';
15162: Function ThemeControl( AControl : TControl) : Boolean';
15163: Function UnthemedDesigner( AControl : TControl) : Boolean';
15164: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15165: begin
15166:   Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String';
15167: end;
15168: Unit uPSC_menus;
15169: Function StripHotkey( const Text : string) : string';
15170: Function GetHotkey( const Text : string) : string';
15171: Function AnsiSameCaption( const Text1, Text2 : string) : Boolean';
15172: Function IsAltGRPressed : boolean';
15173:
15174: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15175: begin
15176:   TCommandEvent', 'Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15177:   SIRegister_IdIMAP4Server(CL);
15178: end;
15179:
15180: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15181: begin
15182:   'HASH_SIZE','LongInt'( 256);
15183:   CL.FindClass('TOBJECT'), 'EVariantSymbolTable');

```

```

15184: CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15185: //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15186: CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15187:   : Integer; Value : Variant; end');
15188: //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15189: SIRegister_TVariantSymbolTable(CL);
15190: end;
15191;
15192: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15193: begin
15194:   SIRegister_TThreadLocalVariables(CL);
15195:   Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD ) : Pchar');
15196: //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD ) : PISC_QUAD';
15197:   Function ThreadLocals : TThreadLocalVariables');
15198:   Procedure WriteDebug( sz : String );
15199:   CL.AddConstantN('UDF_SUCCESS','LongInt'( 0 );
15200:   'UDF_FAILURE','LongInt'( 1 );
15201:   'cSignificantlyLarger','LongInt'( 1024 * 4 );
15202:   CL.AddTypeS('mTByteArray', 'array of byte');
15203:   function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15204:   function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15205:   procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15206:   function IsNetworkConnected: Boolean;
15207:   function IsInternetConnected: Boolean;
15208:   function IsCOMConnected: Boolean;
15209:   function IsNetworkOn: Boolean;
15210:   function IsInternetOn: Boolean;
15211:   function IsCOMON: Boolean;
15212:   Function SetTimer(hWnd : HWND; nIDEEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15213:   TmrProc', 'procedure TmrProc(hWnd: HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);');
15214:   Function SetTimer2( hWnd : HWND; nIDEEvent, uElapse : UINT; lpTimerFunc : TmrProc ) : UINT';
15215:   Function KillTimer( hWnd : HWND; uIDEEvent : UINT ) : BOOL';
15216:   Function wIsWindowUnicode( hWnd : HWND ) : BOOL';
15217:   Function wEnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL';
15218:   Function wIsWindowEnabled( hWnd : HWND ) : BOOL';
15219:   Function GetMenu( hWnd : HWND ) : HMENU';
15220:   Function SetMenu( hWnd : HWND; hMenu : HMENU ) : BOOL';
15221: end;
15222;
15223: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15224: begin
15225:   SIRegister_IDataBlock(CL);
15226:   SIRegister_ISendDataBlock(CL);
15227:   SIRegister_ITransport(CL);
15228:   SIRegister_TDataBlock(CL);
15229:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15230:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15231:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15232:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15233:   SIRegister_TCustomDataBlockInterpreter(CL);
15234:   SIRegister_ISendDataBlock(CL);
15235:   'CallSig','LongWord')( $D800);
15236:   'ResultSig','LongWord')( $D400);
15237:   'asMask','LongWord')( $00FF);
15238:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15239:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15240:   Procedure CheckSignature( Sig : Integer );
15241: end;
15242;
15243: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15244: begin
15245:   //CL.AddTypeS('HINTERNET', '__Pointer');
15246:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15247:   CL.AddTypeS('HINTERNET', 'Integer');
15248:   CL.AddTypeS('HINTERNET2', '__Pointer');
15249:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15250:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15251:   CL.AddTypeS('INTERNET_PORT', 'Word');
15252:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15253:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15254:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15255:   'INTERNET_RFC1123_FORMAT','LongInt'( 0 );
15256:   'INTERNET_RFC1123_BUFSIZE','LongInt'( 30 );
15257:   Function InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var
15258:   lpUrlComponents:TURLComponents):BOOL;
15259:   Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
15260:   dwUrlLength:DWORD):BOOL;
15261:   Function InternetCloseHandle(hInet : HINTERNET) : BOOL';
15262:   Function
15263:     InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
15264:     lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15265:     Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15266:     ;dwContext:DWORD):HINTERNET;
15267:     Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
15268:     lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15269:     Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
15270:     dwContext:DWORD):BOOL;
15271:     Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE ) : BOOL';
15272:   Function
15273:     InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;

```

```

15267: Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD ) : DWORD';
15268: Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD ) : BOOL';
15269: Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15270: Function InternetAutodialHangup( dwReserved : DWORD ) : BOOL';
15271: Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15272: Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15273: Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
lpdwBufferLength : Function InternetCloseHandle( hInet : INTERNET ; : BOOL');
15274: Function InternetConnect(hInet:INTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):INTERNET;
15275: Function InternetQueryDataAvailable(hFile:INTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
dwContext:DWORD):BOOL;
15277: Function FtpFindFirstFile( hConnect : INTERNET; lpszSearchFile : PChar; var lpFindFileData :
TWin32FindData; dwFlags : DWORD; dwContext : DWORD ) : INTERNET';
15278: Function WFtpGetFile( hConnect : INTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD ) : BOOL';
15279: Function
WFtpPutFile(hConct:INTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15280: Function FtpDeleteFile( hConnect : INTERNET; lpszFileName : PChar ) : BOOL';
15281: Function FtpRenameFile(hConnect:INTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15282: Function
FtpOpenFile(hConnect:INTERNET;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):INTERNET;
15283: Function FtpCreateDirectory( hConnect : INTERNET; lpszDirectory : PChar ) : BOOL';
15284: Function FtpRemoveDirectory( hConnect : INTERNET; lpszDirectory : PChar ) : BOOL';
15285: Function FtpSetCurrentDirectory(hConnect:INTERNET; lpszDirectory : PChar ) : BOOL';
15286: Function FtpGetCurrentDirectory(hConnect:INTERNET;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15287: Function
FtpCommand(hConnect:INTERNET;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15288: Function IS_GOPHER_FILE( GopherType : DWORD ) : BOOL';
15289: Function IS_GOPHER_DIRECTORY( GopherType : DWORD ) : BOOL';
15290: Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD ) : BOOL';
15291: Function IS_GOPHER_ERROR( GopherType : DWORD ) : BOOL';
15292: Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD ) : BOOL';
15293: Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD ) : BOOL';
15294: Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD ) : BOOL';
15295: Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD ) : BOOL';
15296: Function IS_GOPHER_ASK( GopherType : DWORD ) : BOOL';
15297: Function IS_GOPHER_PLUS( GopherType : DWORD ) : BOOL';
15298: Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD ) : BOOL';
15299: Function
GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15300: Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL';
15301: Function
GopherOpenFile(hConect:INTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):INTERNET;
15302: Function HttpOpenRequest( hConnect:INTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):INTERNET;
15303: Function
HttpAddRequestHeaders( hReg:INTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15304: Function HttpSendRequest( hRequest: INTERNET; lpszHeaders : PChar;
dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15305: Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15306: Function InternetAttemptConnect( dwReserved : DWORD ) : DWORD';
15307: Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15308: Function InternetErrorDlg(hWnd:HWND;hRequest:INTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15309: Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL ) : DWORD';
15310: Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject ) : Int64';
15311: Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject ) : Bool';
15312: Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
lpFirstCacheEntryInfo:TIInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15313: Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TIInternetCacheEntryInfo; var
lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15314: Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15315: Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15316: Function
InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15317: Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD ) : BOOL';
15318: end;
15319:
15320: procedure SIRegister_Wwstr(CL: TPPSPascalCompiler);
15321: begin
15322: AddTypeS('str CharSet', 'set of char');
15323: TwwGetWordOption(,'(wwgSkipLeadingBlanks, wwgQuotesAsWords,wwgwStripQuotes , wwgSpacesInWords);
15324: AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15325: Procedure strBreakApart( s : string; delimiter : string; parts : TStrings)');
15326: Function strGetToken( s : string; delimiter : string; var APos : integer) : string');
15327: Procedure strStripPreceding( var s : string; delimiter : str CharSet)');
15328: Procedure strStripTrailing( var s : string; delimiter : str CharSet)');
15329: Procedure strStripWhiteSpace( var s : string)');
15330: Function strRemoveChar( str : string; removeChar : char) : string');
15331: Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15332: Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15333: Function wwEqualStr( s1, s2 : string) : boolean');
15334: Function strCount( s : string; delimiter : char) : integer');
15335: Function strWhiteSpace : str CharSet');
15336: Function wwExtractFileNameOnly( const FileName : string) : string');
15337: Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:str CharSet):string;
15338: Function strTrailing( s : string; delimiter : char) : string');

```

```

15339: Function strPreceding( s : string; delimiter : char ) : string');
15340: Function wwstrReplace( s, Find, Replace : string ) : string');
15341: end;
15342:
15343: procedure SIRegister_DataBkr(CL: TPSPPascalCompiler);
15344: begin
15345:   SIRegister_TRemoteDataModule(CL);
15346:   SIRegister_TCRemoteDataModule(CL);
15347: Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)' );
15348: Procedure UnregisterPooled( const ClassID : string)' );
15349: Procedure EnableSocketTransport( const ClassID : string)' );
15350: Procedure DisableSocketTransport( const ClassID : string)' );
15351: Procedure EnableWebTransport( const ClassID : string)' );
15352: Procedure DisableWebTransport( const ClassID : string)' );
15353: end;
15354:
15355: procedure SIRegister_Mathbox(CL: TPSPPascalCompiler);
15356: begin
15357:   Function mxArcCos( x : Real ) : Real';
15358:   Function mxArcSin( x : Real ) : Real';
15359:   Function Comp2Str( N : Comp ) : String';
15360:   Function Int2StrPad0( N : LongInt; Len : Integer ) : String';
15361:   Function Int2Str( N : LongInt ) : String';
15362:   Function mxIsEqual( R1, R2 : Double ) : Boolean';
15363:   Function LogXY( x, y : Real ) : Real';
15364:   Function Pennies2Dollars( C : Comp ) : String';
15365:   Function mxPower( X : Integer; Y : Integer ) : Real';
15366:   Function Real2Str( N : Real; Width, Places : integer ) : String';
15367:   Function mxStr2Comp( MyString : string ) : Comp');
15368:   Function mxStr2Pennies( S : String ) : Comp');
15369:   Function Str2Real( MyString : string ) : Real');
15370:   Function XToTheY( x, y : Real ) : Real');
15371: end;
15372:
15373: //*****Cindy Functions!*****
15374: procedure SIRegister_cyIndy(CL: TPSPPascalCompiler);
15375: begin
15376:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml',
15377:     +' _Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach,
15378:     +' cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15379:   MessagePlainText', 'String 'text/plain');
15380:   CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15381:   MessageAlterText_Html', 'String 'multipart/alternative');
15382:   MessageHtml_Attach', 'String 'multipart/mixed');
15383:   MessageHtml_RelatedAttach', 'String 'multipart/related; type="text/html"');
15384:   MessageAlterText_Html_Attach', 'String 'multipart/mixed');
15385:   MessageAlterText_Html_RelatedAttach', 'String')('multipart/related;type="multipart/alternative"');
15386:   MessageAlterText_Html_Attach_RelatedAttach', 'String 'multipart/mixed');
15387:   MessageReadNotification', 'String')('multipart/report; report-type="disposition-notification"');
15388:   Function ForceDecodeHeader( aHeader : String ) : String');
15389:   Function Base64_EncodeString( Value : String; const aEncoding : TEncoding ) : string');
15390:   Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding ) : String');
15391:   Function Base64_DecodeToBytes( Value : String ) : TBytes');
15392:   Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15393:   Function Get_MD5( const aFileName : string ) : string');
15394:   Function Get_MD5FromString( const aString : string ) : string');
15395: end;
15396:
15397: procedure SIRegister_cySysUtils(CL: TPSPPascalCompiler);
15398: begin
15399:   Function IsFolder( SRec : TSearchrec ) : Boolean');
15400:   Function isFolderReadOnly( Directory : String ) : Boolean');
15401:   Function DirectoryIsEmpty( Directory : String ) : Boolean');
15402:   Function DirectoryWithSubDir( Directory : String ) : Boolean');
15403:   Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15404:   Function DiskFreeBytes( Drv : Char ) : Int64');
15405:   Function DiskBytes( Drv : Char ) : Int64');
15406:   Function GetFileBytes( Filename : String ) : Int64');
15407:   Function GetFilesBytes( Directory, Filter : String ) : Int64');
15408:   SE_CREATE_TOKEN_NAME', 'String 'SeCreateTokenPrivilege');
15409:   SE_ASSIGNPRIMARYTOKEN_NAME', 'String 'SeAssignPrimaryTokenPrivilege');
15410:   SE_LOCK_MEMORY_NAME', 'String 'SeLockMemoryPrivilege');
15411:   SE_INCREASE_QUOTA_NAME', 'String 'SeIncreaseQuotaPrivilege');
15412:   SE_UNSOLICITED_INPUT_NAME', 'String 'SeUnsolicitedInputPrivilege');
15413:   SE_MACHINE_ACCOUNT_NAME', 'String 'SeMachineAccountPrivilege');
15414:   SE_TCB_NAME', 'String 'SeTcbPrivilege');
15415:   SE_SECURITY_NAME', 'String 'SeSecurityPrivilege');
15416:   SE_TAKE_OWNERSHIP_NAME', 'String 'SeTakeOwnershipPrivilege');
15417:   SE_LOAD_DRIVER_NAME', 'String 'SeLoadDriverPrivilege');
15418:   SE_SYSTEM_PROFILE_NAME', 'String 'SeSystemProfilePrivilege');
15419:   SE_SYSTEMTIME_NAME', 'String 'SeSystemtimePrivilege');
15420:   SE_PROF_SINGLE_PROCESS_NAME', 'String 'SeProfileSingleProcessPrivilege');
15421:   SE_INC_BASE_PRIORITY_NAME', 'String 'SeIncreaseBasePriorityPrivilege');
15422:   SE_CREATE_PAGEFILE_NAME', 'String 'SeCreatePagefilePrivilege');
15423:   SE_CREATE_PERMANENT_NAME', 'String 'SeCreatePermanentPrivilege');
15424:   SE_BACKUP_NAME', 'String 'SeBackupPrivilege');
15425:   SE_RESTORE_NAME', 'String 'SeRestorePrivilege');
15426:   SE_SHUTDOWN_NAME', 'String 'SeShutdownPrivilege');
15427:   SE_DEBUG_NAME', 'String 'SeDebugPrivilege');
```

```

15428: SE_AUDIT_NAME', 'String 'SeAuditPrivilege');
15429: SE_SYSTEM_ENVIRONMENT_NAME', 'String 'SeSystemEnvironmentPrivilege');
15430: SE_CHANGE_NOTIFY_NAME', 'String 'SeChangeNotifyPrivilege');
15431: SE_REMOTE_SHUTDOWN_NAME', 'String 'SeRemoteShutdownPrivilege');
15432: SE_UNDOCK_NAME', 'String 'SeUndockPrivilege');
15433: SE_SYNC_AGENT_NAME', 'String 'SeSyncAgentPrivilege');
15434: SE_ENABLE_DELEGATION_NAME', 'String 'SeEnableDelegationPrivilege');
15435: SE_MANAGE_VOLUME_NAME', 'String 'SeManageVolumePrivilege');
15436: end;
15437:
15438:
15439: procedure SIRegister_cyWinUtils(CL: TPPascalCompiler);
15440: begin
15441:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15442:     +'Me, wvWinNT3, wvWinNT4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15443:   Function ShellGetExtensionName( FileName : String ) : String';
15444:   Function ShellGetIconIndex( FileName : String ) : Integer';
15445:   Function ShellGetIconHandle( FileName : String ) : HIcon';
15446:   Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15447:   Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
15448:   Procedure ShellRenameDir( DirFrom, DirTo : string');
15449:   Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15450:   Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15451:   Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String );
15452:   Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string );
15453:   Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
WaitCloseCompletion : boolean) : Boolean';
15454:   Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15455:   Function RemoveDuplicatedPathDelimiter( Str : String ) : String';
15456:   Function cyFileTimeToDate( _FT : TFileTime ) : TDateTime';
15457:   Function GetModificationDate( Filename : String ) : TDateTime';
15458:   Function GetCreationDate( Filename : String ) : TDateTime';
15459:   Function GetLastAccessDate( Filename : String ) : TDateTime';
15460:   Function FileDelete( Filename : String ) : Boolean';
15461:   Function FileIsOpen( Filename : string ) : boolean';
15462:   Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15463:   Function DirectoryDelete( Directory : String ) : Boolean';
15464:   Function GetPrinters( PrintersList : TStrings ) : Integer';
15465:   Procedure SetDefaultPrinter( PrinterName : String );
15466:   Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15467:   Function WinToDosPath( WinPathName : String ) : String';
15468:   Function DosToWinPath( DosPathName : String ) : String';
15469:   Function cyGetWindowsVersion : TWindowsVersion';
15470:   Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean ) : Boolean';
15471:   Procedure WindowsShutDown( Restart : boolean );
15472:   Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
FileIcon:string;NumIcone:integer);
15473:   Procedure GetWindowsFonts( FontsList : TStrings );
15474:   Function GetAvailableFilename( DesiredFileName : String ): String';
15475: end;
15476:
15477: procedure SIRegister_cyStrUtils(CL: TPPascalCompiler);
15478: begin
15479:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15480:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15481:   Type(TStringRead', '( srFromLeft, srFromRight )');
15482:   Type(TStringReads', 'set of TStringRead');
15483:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15484:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15485:   Type(TWordsOptions', 'set of TWordsOption');
15486:   Type(TCarType', '( ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15487:   Type(TCarTypes', 'set of TCarType');
15488:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15489:   CarTypeAlphabetic'': LongInt):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15490:   Function Char_GetType( aChar : Char ) : TCarType';
15491:   Function SubString_Count( Str : String; Separator : Char ) : Integer';
15492:   Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15493:   Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word ) : String';
15494:   Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15495:   Procedure SubString_Add( var Str : String; Separator : Char; Value : String );
15496:   Procedure SubString_Insert(var Str:String;Separator:Char;SubStringIndex:Word; Value : String );
15497:   Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String );
15498:   Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean';
15499:   Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15500:   Function SubString_Ribbon( Str : string; Separator : Char; Current : Word; MoveBy : Integer ): Integer';
15501:   Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String';
15502:   Function String_Quote( Str : String ) : String';
15503:   Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char ) : Char';
15504:   Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15505:   Function String_GetWord( Str : String; StringRead : TStringRead ) : String';
15506:   Function String_GetInteger( Str : String; StringRead : TStringRead ) : string';
15507:   Function String_ToInt( Str : String ) : Integer';
15508:   Function String_Uppercase( Str : String; Options : TWordsOptions ) : String';
15509:   Function String_Lowercase( Str : String; Options : TWordsOptions ) : String';
15510:   Function String_Reverse( Str : String ) : String';
15511:   Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15512:   Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
TCaseSensitive):Integer';
15513:   Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer ) : String';

```

```

15514: Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15515: Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
  Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String');
15516: Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer ) : String');
15517: Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String');
15518: Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15519: Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15520: Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String';
15521: Function String_End( Str : String; Cars : Word) : String');
15522: Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
  AlwaysFindFromBeginning : Boolean) : String');
15523: Function String_SubstCar( Str : String; Old, New : Char ) : String');
15524: Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15525: Function String_SameCars(Str1,Str2:String;StopCount:_IfDifferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15526: Function String_IsNumbers( Str : String) : Boolean');
15527: Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15528: Function StringToCsvCell( aStr : String) : String');
15529: end;
15530:
15531: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15532: begin
15533:   Function LongDayName( aDate : TDate ) : String';
15534:   Function LongMonthName( aDate : TDate ) : String');
15535:   Function ShortYearOf( aDate : TDate ) : byte';
15536:   Function DateToStrYYYYMMDD( aDate : TDate ) : String');
15537:   Function StrYYYYMMDDToDate( aStr : String ) : TDate';
15538:   Function SecondsToMinutes( Seconds : Integer ) : Double';
15539:   Function MinutesToSeconds( Minutes : Double ) : Integer');
15540:   Function MinutesToHours( Minutes : Integer ) : Double');
15541:   Function HoursToMinutes( Hours : Double ) : Integer');
15542:   Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String ) : TDateTime');
15543:   Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer );
15544:   Function MergeDateWithTime( aDate : TDate; aTime : TDateTime ) : TDateTime');
15545:   Function GetMinutesBetween( DateTimel, DateTime2 : TDateTime ) : Int64';
15546:   Function GetMinutesBetween(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15547:   Function GetSecondsBetween( DateTimel, DateTime2 : TDateTime ) : Int64');
15548:   Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
  RsltBegin:TDateTime; RsltEnd : TDateTime ) : Boolean';
15549:   Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15550:   Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime ) : Boolean');
15551: end;
15552:
15553: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15554: begin
15555:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15556:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15557:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15558:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15559:   Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions ) : Integer';
15560:   Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind ) : Integer';
15561:   Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer';
15562:   Procedure StringsReplace(aList:TStrings;OldStr:String;NewStr:String;ValueKind : TStringsValueKind)');
15563:   Procedure StringsSort( aList : TStrings; SortType : TStringsSortType );
15564:   Function TreeNodeLocate( ParentNode : TTreeNode; Value : String ) : TTreeNode';
15565:   Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String ) : TTreeNode';
15566:   Function
    TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode';
15567:   Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer ) : TTreeNode';
15568:   Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
  CopySubChildren:Bool;
    Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String ) );
15569:   Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15570:   Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint ) : TControl';
15571:   Procedure cyCenterControl( aControl : TControl );
15572:   Function GetLastParent( aControl : TControl ) : TWinControl';
15573:   Function GetControlBitmap( aControl : TWinControl ) : TBitmap';
15574:   Function GetRichEditBitmap( aRichEdit : TRichEdit ) : TBitmap';
15575:   Function GetRichEditBitmap( aRichEdit : TRichEdit ) : TBitmap';
15576: end;
15577:
15578: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15579: begin
15580:   Function TablePackTable( Tab : TTable ) : Boolean';
15581:   Function TableRegenIndexes( Tab : TTable ) : Boolean';
15582:   Function TableShowDeletedRecords( Tab : TTable; Show : Boolean ) : Boolean';
15583:   Function TableUndeleteRecord( Tab : TTable ) : Boolean';
15584:   Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15585:   Function TableDeleteIndex( Tab : TTable; IndexFieldName : String ) : Boolean';
15586:   Function TableEmptyTable( Tab : TTable ) : Boolean';
15587:   Function TableFindKey( aTable : TTable; Value : String ) : Boolean';
15588:   Procedure TableFindNearest( aTable : TTable; Value : String );
15589:   Function
    TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Bool):TTable;
15590:   Function
    TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15591:   Function DateToBDESQLODate( aDate : TDate; const DateFormat : String ) : String';
15592: end;
15593:
15594: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15595: begin

```

```

15596: SIRегистер_TcyRunTimeDesign(CL);
15597: SIRегистер_TcyShadowText(CL);
15598: SIRегистер_TcyBgPicture(CL);
15599: SIRегистер_TcyGradient(CL);
15600: SIRегистер_TcyBevel(CL);
15601: //CL.AddTypes('TcyBevelClass', 'class of tcyBevel');
15602: SIRегистер_TcyBevels(CL);
15603: SIRегистер_TcyImagelistOptions(CL);
15604: Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture );
15605: end;
15606:
15607: procedure SIRегистер_cyGraphics(CL: TPPascalCompiler);
15608: begin
15609:   Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
Maxdegrade : Byte ); +
15610:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap );
15611:   Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15612:   Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15613:   Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad : Byte; toRect : TRect; OrientationShape : TDgradOrientationShape );
15614:   Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad : Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap );
15615:   Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer );
15616:   Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer );
15617:   Procedure cyFrameL( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
BottomColor : TColor; const Width : Integer; const RoundRect : boolean );
15618:   Procedure cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,
BottomRightColor:TColor;Width:Integer;const DrawLeft:Boolean;const DrawTop:Boolean;const DrawRight:Boolean;const
DrawBottom:Boolean;const RoundRect:bool);
15619:   Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
aState : TButtonState; Focused, Hot : Boolean );
15620:   Procedure cyDrawButton(Canvas:TCanvas;Caption:String;ARect:TRect;GradientColor1,GradientColor2:TColor;
aState : TButtonState; Focused, Hot : Boolean );
15621:   Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
aState : TButtonState; Focused, Hot : Boolean );
15622:   Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean );
15623:   Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor );
15624:   Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer );
15625:   Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
TAlignment;Layout:TTextLayout;WordWrap:Boolean):LongInt;
15626:   Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
WordWrap : Boolean; CaptionRender : TCaptionRender ) : LongInt );
15627:   Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt );
15628:   Function cyCreateFontIndirect( fromFont : TFont; Angle : Double ) : TFont ;
15629:   Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation ) : TFont ;
15630:   Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout );
15631:   Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone : Integer;
const OnlyCalcFoldLine : Boolean ) : TLineCoord ;
15632:   Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone : Integer;
const OnlyCalcFoldLine : Boolean ) : TLineCoord ;
15633:   Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint ) : boolean ;
15634:   Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint ) : boolean ;
15635:   Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint ) : boolean ;
15636:   Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint ) : boolean ;
15637:   Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect );
15638:   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);
15639:   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 );
15640:   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 );
15641:   Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap );
15642:   Function ValidGraphic( aGraphic : TGraphic ) : Boolean ;
15643:   Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer ) : TColor ;
15644:   Function ColorModify( Color : TColor; incR, incG, incB : Integer ) : TColor ;
15645:   Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer ) : TColor ;
15646:   Function ColorSetPercentPale( Color : TColor; IncPercent : integer ) : TColor ;
15647:   Function MediumColor( Color1, Color2 : TColor ) : TColor ;
15648:   Function ClientToScreenRect( aControl : TControl; aControlRect : TRect ) : TRect ;
15649:   Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect ) : TRect ;
15650:   Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect ) : TRect ;
15651:   Procedure InflateRectPercent( var aRect : TRect; withPercent : Double );
15652:   Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double ) : TRect ;
15653:   Function GetProportionalRect( fromRect, InsideRect : TRect ) : TRect ;
15654:   Function PointInRect( const aPt : TPoint; const aRect : TRect ) : boolean ;
15655:   Function PointInEllispe( const aPt : TPoint; const aRect : TRect ) : boolean ;
15656:   Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String ) : Integer ;
15657: end;
15658:

```

```

15659: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15660: begin
15661:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15662:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15663:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15664:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15665:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15666:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,
15667:     +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft )');
15668:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15669:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15670:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15671:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15672:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bnReverseFromColor,bmInvertReverse,
15673:     bmInvertReverseFromColor);
15674:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
15675:     rjResizeTopLeft, rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15676:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15677:   cCaptionOrientationWarning', 'String')('Note that text orientation doesn''t work with all fonts!');
15678: end;
15679:
15680: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15681: begin
15682:   Const SERVICES_ACTIVE_DATABASEA', 'String' 'ServicesActive');
15683:   SERVICES_ACTIVE_DATABASEW', 'String') 'ServicesActive');
15684:   Const SERVICES_FAILED_DATABASEA', 'String' 'ServicesFailed');
15685:   Const SERVICES_FAILED_DATABASEW', 'String' 'ServicesFailed');
15686:   Const SC_GROUP_IDENTIFIERA', 'String'). '+');
15687:   Const SC_GROUP_IDENTIFIERW', 'String') '+');
15688:   Const SC_GROUP_IDENTIFIER', 'string 'SC_GROUP_IDENTIFIERA');
15689:   Const SERVICE_NO_CHANGE', 'LongWord $FFFFFFF';
15690:   Const SERVICE_ACTIVE', 'LongWord')($00000001);
15691:   Const SERVICE_INACTIVE', 'LongWord $00000002);
15692:   Const SERVICE_CONTROL_STOP', 'LongWord $00000001);
15693:   Const SERVICE_CONTROL_PAUSE', 'LongWord $00000002);
15694:   Const SERVICE_CONTROL_CONTINUE', 'LongWord $00000003);
15695:   Const SERVICE_CONTROL_INTERROGATE', 'LongWord $00000004);
15696:   Const SERVICE_CONTROL_SHUTDOWN', 'LongWord $00000005);
15697:   Const SERVICE_STOPPED', 'LongWord $00000001);
15698:   Const SERVICE_START_PENDING', 'LongWord $00000002);
15699:   Const SERVICE_STOP_PENDING', 'LongWord $00000003);
15700:   Const SERVICE_RUNNING', 'LongWord $00000004);
15701:   Const SERVICE_CONTINUE_PENDING', 'LongWord $00000005);
15702:   Const SERVICE_PAUSED', 'LongWord $00000006);
15703:   Const SERVICE_PAUSED', 'LongWord $00000007);
15704:   Const SERVICE_ACCEPT_STOP', 'LongWord $00000001);
15705:   Const SERVICE_ACCEPT_PAUSE_CONTINUE', 'LongWord $00000002);
15706:   Const SERVICE_ACCEPT_SHUTDOWN', 'LongWord $00000004);
15707:   Const SC_MANAGER_CONNECT', 'LongWord $0001);
15708:   Const SC_MANAGER_CREATE_SERVICE', 'LongWord $0002;
15709:   Const SC_MANAGER_ENUMERATE_SERVICE', 'LongWord $0004);
15710:   Const SC_MANAGER_LOCK', 'LongWord $0008);
15711:   Const SC_MANAGER_QUERY_LOCK_STATUS', 'LongWord $0010);
15712:   Const SC_MANAGER_MODIFY_BOOT_CONFIG', 'LongWord $0020);
15713:   Const SERVICE_QUERY_CONFIG', 'LongWord $0001);
15714:   Const SERVICE_CHANGE_CONFIG', 'LongWord $0002);
15715:   Const SERVICE_QUERY_STATUS', 'LongWord $0004);
15716:   Const SERVICE_ENUMERATE_DEPENDENTS', 'LongWord $0008);
15717:   Const SERVICE_START', 'LongWord $0010);
15718:   Const SERVICE_STOP', 'LongWord $0020);
15719:   Const SERVICE_PAUSE_CONTINUE', 'LongWord $0040);
15720:   Const SERVICE_INTERROGATE', 'LongWord $0080);
15721:   Const SERVICE_USER_DEFINED_CONTROL', 'LongWord $0100);
15722:   Const SERVICE_KERNEL_DRIVER', 'LongWord $00000001);
15723:   Const SERVICE_FILE_SYSTEM_DRIVER', 'LongWord $00000002);
15724:   Const SERVICE_ADAPTER', 'LongWord $00000004);
15725:   Const SERVICE_RECOGNIZER_DRIVER', 'LongWord $00000008);
15726:   Const SERVICE_WIN32_OWN_PROCESS', 'LongWord $00000010);
15727:   Const SERVICE_WIN32_SHARE_PROCESS', 'LongWord $00000020);
15728:   Const SERVICE_INTERACTIVE_PROCESS', 'LongWord $00000100);
15729:   Const SERVICE_BOOT_START', 'LongWord $00000000);
15730:   Const SERVICE_SYSTEM_START', 'LongWord $00000001);
15731:   Const SERVICE_AUTO_START', 'LongWord $00000002);
15732:   Const SERVICE_DEMAND_START', 'LongWord $00000003);
15733:   Const SERVICE_DISABLED', 'LongWord $00000004);
15734:   Const SERVICE_ERROR_IGNORE', 'LongWord $00000000);
15735:   Const SERVICE_ERROR_NORMAL', 'LongWord $00000001);
15736:   Const SERVICE_ERROR_SEVERE', 'LongWord $00000002);
15737:   Const SERVICE_ERROR_CRITICAL', 'LongWord $00000003);
15738:   CL.AddTypeS('SC_HANDLE', 'THandle');
15739: //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15740:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15741:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState :
15742:     +' : DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe
15743:     +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15744:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15745:   Const TServiceStatus', '_SERVICE_STATUS');

```

```

15746: CL.AddTypeS(' _ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15747:   +'playName : PChar; ServiceStatus : TServiceStatus; end');
15748: ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15749: _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15750: TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15751: TEnumServiceStatus', 'TEnumServiceStatusA');
15752: SC_LOCK', '__Pointer');
15753: _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOwner: PChar; dwLockDuration:DWORD; end';
15754: _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15755: QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15756: QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15757: TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15758: //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15759: TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15760: _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15761:   +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15762:   +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15763:   +'iceStartName : PChar; lpDisplayname : PChar; end');
15764: _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15765: QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15766: QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15767: TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15768: TQueryServiceConfig', 'TQueryServiceConfigA');
15769: Function CloseServiceHandle( hSCObject : SC_HANDLE ) : BOOL';
15770: Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15771: Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
  dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;
  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar ) : SC_HANDLE';
15772: Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
  dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;
  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar ) : SC_HANDLE';
15773: Function DeleteService( hService : SC_HANDLE ) : BOOL';
15774: Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
  TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL';
15775: Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
  lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
  : DWORD):BOOL';
15776: Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
  lpcchBuffer:DWORD):BOOL';
15777: Function GetService DisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayname:PChar;var
  lpcchBuffer:DWORD):BOOL;
15778: Function LockServiceDatabase( hSCManager : SC_HANDLE ) : SC_LOCK';
15779: Function NotifyBootConfigStatus( BootAcceptable : BOOL ) : BOOL';
15780: Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD ) : SC_HANDLE';
15781: Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE';
15782: Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
  cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : BOOL';
15783: Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15784: Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15785: Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15786: Function UnlockServiceDatabase( ScLock : SC_LOCK ) : BOOL';
15787: end;
15788: end;
15789: end;
15790: 
15791: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15792: begin
15793:   Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
  AStartOfWeek : TDayOfWeekName; AWeekends : TDaysOfWeek; AWeekendColor : TColor;
  BtnHints : TStrings; MinDate : TDateTime; MaxDate : TDateTime ) : Boolean;
15794:   Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
  DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDateTime;
  MaxDate:TDateTime):Boolean;
15795:   Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime) : Boolean;
15796:   Function CreatePopupCalendar(AOwner : TComponent; ABiDiMode : TBiDiMode; MinDate : TDateTime; MaxDate : TDateTime) :
  TWinControl;
15797:   Procedure SetupPopupCalendar(PopupCalendar:TWinControl; AStartOfWeek : TDayOfWeekName; AWeekends : TDaysOfWeek;
  AWeekendColor : TColor; BtnHints : TStrings; FourDigitYear : Boolean; MinDate : TDateTime; MaxDate : TDateTime );
15798:   Function CreateNotifyThread(const
  FolderName : string; WatchSubtree : Bool; Filter : TFileChangeFilters) : TJvNotifyThread;
15799: end;
15800: 
15801: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15802: begin
15803:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EJclNtfsError');
15804:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15805:   Function NtfsGetCompression2( const FileName : TFileName; var State : Short ) : Boolean';
15806:   Function NtfsGetCompression12( const FileName : TFileName ) : TFileCompressionState';
15807:   Function NtfsSetCompression2( const FileName : TFileName; const State : Short ) : Boolean';
15808:   Procedure NtfsSetfileCompression2(const FileName : TFileName; const State : TFileCompressionState)';
15809:   Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)';
15810:   Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)';
15811:   Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)';
15812:   Function NtfsSetSparse2( const FileName : string ) : Boolean';
15813:   Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64 ) : Boolean';
15814:   Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64 ) : Boolean';
15815:   Function NtfsSparseStreamsSupported2( const Volume : string ) : Boolean';
15816:   Function NtfsGetSparse2( const FileName : string ) : Boolean';
15817:   Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD ) : Boolean';
15818:   Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword ) : Boolean';
15819:   Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD ) : Boolean';
15820:   Function NtfsReparsePointsSupported2( const Volume : string ) : Boolean';

```

```

15821: Function NtfsFileHasReparsePoint2( const Path : string ) : Boolean';
15822: Function NtfsIsFolderMountPoint2( const Path : string ) : Boolean';
15823: Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char ) : Boolean';
15824: Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString ) : Boolean';
15825: CL.AddTypeS('TOpLock', '( olExclusive, olReadonly, olBatch, olFilter )');
15826: Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15827: Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15828: Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15829: Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15830: Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped ):Boolean';
15831: Function NtfsCreateJunctionPoint2( const Source, Destination : string ) : Boolean';
15832: Function NtfsDeleteJunctionPoint2( const Source : string ) : Boolean';
15833: Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15834: CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15835: +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15836: CL.AddTypeS('TStreamIds', 'set of TStreamId');
15837: TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15838: CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15839: +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15840: Function NtfsGetFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15841: Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean';
15842: Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean';
15843: Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String ) : Boolean';
15844: Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : Ansistring ) : Boolean';
15845: Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString ) : Boolean';
15846: CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15847: Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo ) : Boolean';
15848: Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card:const
List:TStrings):Bool
15849: Function NtfsDeleteHardLinks2( const FileName : string ) : Boolean';
15850: FindClass ('TOBJECT','EJclFileSummaryError');
15851: TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )';
15852: TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )';
15853: TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID ) : Boolean';
15854: CL.AddClassN(CL.FindClass('TOBJECT','TJclFileSummary'));
15855: SIRegister_TJclFilePropertySet(CL);
15856: //CL.AddTypes('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15857: SIRegister_TJclFileSummary(CL);
15858: SIRegister_TJclDocSummaryInformation(CL);
15859: SIRegister_TJclMediaFileSummaryInformation(CL);
15860: SIRegister_TJclMSISummaryInformation(CL);
15861: SIRegister_TJclShellSummaryInformation(CL);
15862: SIRegister_TJclStorageSummaryInformation(CL);
15863: SIRegister_TJclImageSummaryInformation(CL);
15864: SIRegister_TJclWebViewSummaryInformation(CL);
15865: SIRegister_TJclDisplacedSummaryInformation(CL);
15866: SIRegister_TJclBriefCaseSummaryInformation(CL);
15867: SIRegister_TJclMiscSummaryInformation(CL);
15868: SIRegister_TJclMusicSummaryInformation(CL);
15869: SIRegister_TJclMusicSummaryInformation(CL);
15870: SIRegister_TJclDRMSummaryInformation(CL);
15871: SIRegister_TJclVideoSummaryInformation(CL);
15872: SIRegister_TJclAudioSummaryInformation(CL);
15873: SIRegister_TJclControlPanelSummaryInformation(CL);
15874: SIRegister_TJclVolumeSummaryInformation(CL);
15875: SIRegister_TJclShareSummaryInformation(CL);
15876: SIRegister_TJclLinkSummaryInformation(CL);
15877: SIRegister_TJclQuerySummaryInformation(CL);
15878: SIRegister_TJclImageInformation(CL);
15879: SIRegister_TJclJpegSummaryInformation(CL);
15880: end;
15881:
15882: procedure SIRegister_Jcl8087(CL: TPPSPascalCompiler);
15883: begin
15884: AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15885: T8087Rounding', '( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )';
15886: T8087Infinity', '( icProjective, icAffine )';
15887: T8087Exception', '( emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision';
15888: CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15889: Function Get8087ControlWord : Word';
15890: Function Get8087Infinity : T8087Infinity';
15891: Function Get8087Precision : T8087Precision';
15892: Function Get8087Rounding : T8087Rounding';
15893: Function Get8087StatusWord( ClearExceptions : Boolean ) : Word';
15894: Function Set8087Infinity( const Infinity : T8087Infinity ) : T8087Infinity';
15895: Function Set8087Precision( const Precision : T8087Precision ) : T8087Precision';
15896: Function Set8087Rounding( const Rounding : T8087Rounding ) : T8087Rounding';
15897: Function Set8087ControlWord( const Control : Word ) : Word';
15898: Function ClearPending8087Exceptions : T8087Exceptions';
15899: Function GetPending8087Exceptions : T8087Exceptions';
15900: Function GetMasked8087Exceptions : T8087Exceptions';
15901: Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions';
15902: Function Mask8087Exceptions( Exceptions:T8087Exceptions ) : T8087Exceptions';
15903: Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions';
15904: end;
15905:
15906: procedure SIRegister_JvBoxProcs(CL: TPPSPascalCompiler);
15907: begin
15908: Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl );

```

```

15909: Procedure BoxMoveAllItems( SrcList, DstList : TWinControl );
15910: Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
15911: Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer );
15912: Procedure BoxMoveSelected( List : TWinControl; Items : TStrings );
15913: Procedure BoxSetItem( List : TWinControl; Index : Integer );
15914: Function BoxGetFirstSelection( List : TWinControl ) : Integer ;
15915: Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer ) : Boolean ;
15916: end;
15917:
15918: procedure SIRegister_UrlMon(CL: TPSPPascalCompiler);
15919: begin
15920: //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString('URL Context');
15921: //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString('AsyncCallee');
15922: CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6 );
15923: type ULONG', 'Cardinal';
15924:   LPCWSTR', 'PChar');
15925: CL.AddTypeS('LPWSTR', 'PChar');
15926: LPSTR', 'PChar');
15927: TBindVerb', 'ULONG');
15928: TBindInfoF', 'ULONG');
15929: TBindF', 'ULONG');
15930: TBSDF', 'ULONG');
15931: TBindStatus', 'ULONG');
15932: TCIPStatus', 'ULONG');
15933: TBindString', 'ULONG');
15934: TPiFlags', 'ULONG');
15935: TOIBdgFlags', 'ULONG');
15936: TParseAction', 'ULONG');
15937: TPSUAction', 'ULONG');
15938: TQueryOption', 'ULONG');
15939: TPUAF', 'ULONG');
15940: TSZMFlags', 'ULONG');
15941: TUrlZone', 'ULONG');
15942: TUrlTemplate', 'ULONG');
15943: TZAFlags', 'ULONG');
15944: TUrlZoneReg', 'ULONG');
15945: 'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001 );
15946: CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002 );
15947: const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004 );
15948: const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008 );
15949: const 'CF_NULL','LongInt').SetInt( 0 );
15950: const 'CFSTR_MIME_NULL','LongInt').SetInt( 0 );
15951: const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain' );
15952: const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext' );
15953: const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-bitmap' );
15954: const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript' );
15955: const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff' );
15956: const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic' );
15957: const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav' );
15958: const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav' );
15959: const 'CFSTR_MIME_GIF','String').SetString( 'image/gif' );
15960: const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg' );
15961: const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg' );
15962: const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff' );
15963: const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png' );
15964: const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp' );
15965: const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg' );
15966: const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf' );
15967: const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf' );
15968: const 'CFSTR_MIME_AVI','String').SetString( 'video/avi' );
15969: const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg' );
15970: const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals' );
15971: const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream' );
15972: const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream' );
15973: const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf' );
15974: const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff' );
15975: const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio' );
15976: const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm' );
15977: const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime' );
15978: const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo' );
15979: const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie' );
15980: const 'CFSTR_MIME_HTML','String').SetString( 'text/html' );
15981: const 'MK_S_ASYNCRONOUS','LongWord').SetUInt( $000401E8 );
15982: const 'S_ASYNCRONOUS','LongWord').SetUInt( $000401E8 );
15983: const 'E_PENDING','LongWord').SetUInt( $8000000A );
15984: CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15985: SIRegister_IPersistMoniker(CL);
15986: SIRegister_IBindProtocol(CL);
15987: SIRegister_IBinding(CL);
15988: const 'BINDVERB_GET','LongWord').SetUInt( $00000000 );
15989: const 'BINDVERB_POST','LongWord').SetUInt( $00000001 );
15990: const 'BINDVERB_PUT','LongWord').SetUInt( $00000002 );
15991: const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003 );
15992: const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001 );
15993: const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002 );
15994: const 'BINDF_ASYNCRONOUS','LongWord').SetUInt( $00000001 );
15995: const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002 );
15996: const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004 );

```

```

15997: const 'BINDF_OFFLINEOPERATION', 'LongWord').SetUInt( $00000008);
15998: const 'BINDF_GETNEWESTVERSION', 'LongWord').SetUInt( $00000010);
15999: const 'BINDF_NOWRITECACHE', 'LongWord').SetUInt( $00000020);
16000: const 'BINDF_NEEDFILE', 'LongWord').SetUInt( $00000040);
16001: const 'BINDF_PULLDATA', 'LongWord').SetUInt( $00000080);
16002: const 'BINDF_IGNORESECURITYPROBLEM', 'LongWord').SetUInt( $00000100);
16003: const 'BINDF_RESYNCHRONIZE', 'LongWord').SetUInt( $00000200);
16004: const 'BINDF_HYPERLINK', 'LongWord').SetUInt( $00000400);
16005: const 'BINDF_NO_UI', 'LongWord').SetUInt( $00000800);
16006: const 'BINDF_SILENTOPERATION', 'LongWord').SetUInt( $00001000);
16007: const 'BINDF_PRAGMA_NO_CACHE', 'LongWord').SetUInt( $00002000);
16008: const 'BINDF_FREE_THREADED', 'LongWord').SetUInt( $00010000);
16009: const 'BINDF_DIRECT_READ', 'LongWord').SetUInt( $00020000);
16010: const 'BINDF_FORMS_SUBMIT', 'LongWord').SetUInt( $00040000);
16011: const 'BINDF_GETFROMCACHE_IF_NET_FAIL', 'LongWord').SetUInt( $00080000);
16012: //const 'BINDF_DONTUSECACHE', '').SetString( BINDF_GETNEWESTVERSION);
16013: //const 'BINDF_DONTPUTINCACHE', '').SetString( BINDF_NOWRITECACHE);
16014: //const 'BINDF_NOCOPYDATA', '').SetString( BINDF_PULLDATA);
16015: const 'BSCF_FIRSTDATANOTIFICATION', 'LongWord').SetUInt( $00000001);
16016: const 'BSCF_INTERMEDIATEDATANOTIFICATION', 'LongWord').SetUInt( $00000002);
16017: const 'BSCF_LASTDATANOTIFICATION', 'LongWord').SetUInt( $00000004);
16018: const 'BSCF_DATAFULLYAVAILABLE', 'LongWord').SetUInt( $00000008);
16019: const 'BSCF_AVAILABLEDATASIZEUNKNOWN', 'LongWord').SetUInt( $00000010);
16020: const 'BINDSTATUS_FINDINGRESOURCE', 'LongInt').SetInt( 1);
16021: const 'BINDSTATUS_CONNECTING', 'LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16022: const 'BINDSTATUS_REDIRECTING', 'LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16023: const 'BINDSTATUS_BEGINDOWNLOADDATA', 'LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16024: const 'BINDSTATUS_DOWNLOADINGDATA', 'LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16025: const 'BINDSTATUS_ENDDOWNLOADDATA', 'LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16026: const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS', 'LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
16027: const 'BINDSTATUS_INSTALLINGCOMPONENTS', 'LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
16028: const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS', 'LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
16029: const 'BINDSTATUS_USINGCACHEDCOPY', 'LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16030: const 'BINDSTATUS_SENDINGREQUEST', 'LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16031: const 'BINDSTATUS_CLASSIDAVAILABLE', 'LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16032: const 'BINDSTATUS_MIMETYPEAVAILABLE', 'LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16033: const 'BINDSTATUS_CACHEFILENAMEAVAILABLE', 'LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
16034: const 'BINDSTATUS_BEGINSYNCOPERATION', 'LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
16035: const 'BINDSTATUS_ENDSYNCOPERATION', 'LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
16036: const 'BINDSTATUS_BEGINUPLOADDATA', 'LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16037: const 'BINDSTATUS_UPLOADINGDATA', 'LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16038: const 'BINDSTATUS_ENDUPLOADDATA', 'LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16039: const 'BINDSTATUS_PROTOCOLCLASSID', 'LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16040: const 'BINDSTATUS_ENCODING', 'LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16041: const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE', 'LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16042: const 'BINDSTATUS_CLASSINSTALLLOCATION', 'LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
16043: const 'BINDSTATUS_DECODING', 'LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16044: const 'BINDSTATUS_LOADINGMIMEHANDLER', 'LongInt').SetInt( BINDSTATUS_DECODING + 1);
16045: const 'BINDSTATUS_CONTENTDISPOSITIONATTACH', 'LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
16046: const 'BINDSTATUS_FILTERREPORTMIMETYPE', 'LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
16047: const 'BINDSTATUS_CLSIDCANINSTANTIATE', 'LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
16048: const 'BINDSTATUS_IUNKNOWNAVAILABLE', 'LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16049: const 'BINDSTATUS_DIRECTBIND', 'LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16050: const 'BINDSTATUS_RAWMIMETYPE', 'LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16051: const 'BINDSTATUS_PROXYDETECTING', 'LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16052: const 'BINDSTATUS_ACCEPTRANGES', 'LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16053: const 'BINDSTATUS_COOKIE_SENT', 'LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16054: const 'BINDSTATUS_COMPACT_POLICY_RECEIVED', 'LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16055: const 'BINDSTATUS_COOKIE_SUPPRESSED', 'LongInt').SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED + 1);
16056: const 'BINDSTATUS_COOKIE_STATE_UNKNOWN', 'LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16057: const 'BINDSTATUS_COOKIE_STATE_ACCEPT', 'LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
16058: const 'BINDSTATUS_COOKIE_STATE_REJECT', 'LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16059: const 'BINDSTATUS_COOKIE_STATE_PROMPT', 'LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
16060: const 'BINDSTATUS_COOKIE_STATE_LEASH', 'LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16061: const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE', 'LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
16062: const 'BINDSTATUS_POLICY_HREF', 'LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16063: const 'BINDSTATUS_P3P_HEADER', 'LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16064: const 'BINDSTATUS_SESSION_COOKIE_RECEIVED', 'LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16065: const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED', 'LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
16066: const 'BINDSTATUS_SESSION_COOKIES_ALLOWED', 'LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED + 1);
16067: const 'BINDSTATUS_CACHECONTROL', 'LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16068: const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME', 'LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
16069: const 'BINDSTATUS_MIMETEXTPLAINMISMATCH', 'LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
16070: const 'BINDSTATUS_PUBLISHERAVAILABLE', 'LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
16071: const 'BINDSTATUS_DISPLAYNAMEAVAILABLE', 'LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16072: // PBindInfo , '^TBindInfo // will not work');
16073: {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16074: +'medata : TStgMedium; grfBindInfo : DWORD; dwBindVerb : DWORD; szCustomVe'
16075: +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16076: +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16077: +'+UID; pUnk : IUnknown; dwReserved : DWORD; end';
16078: TBindInfo', '_tagBINDINFO');
16079: BINDINFO', '_tagBINDINFO');
16080: _REMSecurity_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16081: +'criptor : DWORD; bInheritHandle : BOOL; end');
16082: TRemSecurityAttributes', '_REMSecurity_ATTRIBUTES');
16083: REMSecurity_ATTRIBUTES', '_REMSecurity_ATTRIBUTES');
16084: //PREmBindInfo', '^TRemBindInfo // will not work');
16085: {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '

```

```

16086: +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16087: +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16088: +' ; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16089: +'n; dwReserved : DWORD; end');
16090: TRemBindInfo', '_tagRemBINDINFO');
16091: RemBINDINFO', '_tagRemBINDINFO');
16092: //PREmFormatEtc', '^TRemFormatEtc // will not work');
16093: tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16094: TRemFormatEtc', 'tagRemFORMATETC');
16095: RemFORMATETC', 'tagRemFORMATETC');
16096: SIRegister_IBindStatusCallback(CL);
16097: SIRegister_IAuthenticate(CL);
16098: SIRegister_IHttpNegotiate(CL);
16099: SIRegister_IWindowForBindingUI(CL);
16100: const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16101: const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16102: const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16103: const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16104: const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16105: const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16106: const 'CIP_EXE_SELF_REGISTRATION_TIMEOUT','LongInt').SetInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
16107: const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTRATION_TIMEOUT + 1);
16108: const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16109: const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16110: SIRegister_ICodeInstall(CL);
16111: SIRegister_IWInetInfo(CL);
16112: const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16113: SIRegister_IHttpSecurity(CL);
16114: SIRegister_IWInetHttpInfo(CL);
16115: SIRegister_IBindHost(CL);
16116: const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16117: const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16118: const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16119: Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback ) : HResult';
16120: Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback ) : HResult';
16121: Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
: IBindStatusCallback ) : HResult';
16122: Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
IBindStatusCallback ) : HResult';
16123: Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult');
16124: Function HlinkGoBack( unk : IUnknown ) : HResult';
16125: Function HlinkGoForward( unk : IUnknown ) : HResult';
16126: Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR ) : HResult';
16127: // Function HlinkNavigateMoniker( unk : IUnknown; mkTarget : IMoniker ) : HResult');
16128: SIRegister_IInternet(CL);
16129: const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16130: const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16131: const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16132: const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16133: const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16134: const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16135: const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16136: const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16137: const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16138: const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16139: const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16140: const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16141: const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16142: const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16143: //POLEStrArray', '^TOLESTRArray // will not work');
16144: SIRegister_IInternetBindInfo(CL);
16145: const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16146: const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16147: const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16148: const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16149: const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16150: const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16151: const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16152: const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16153: const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16154: const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16155: const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16156: //const 'PI_DOCFILECLSIDLOOKUP','','').SetString( PI_CLSIDLOOKUP);
16157: //PProtocolData', '^TProtocolData // will not work');
16158: _tagPROTOCOLDATA', 'record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16159: TProtocolData', '_tagPROTOCOLDATA');
16160: PROTOCOLDATA', '_tagPROTOCOLDATA');
16161: CL.AddInterface(CL.FindInterface('IUNKNOWN'),IIInternetProtocolSink, 'IIInternetProtocolSink');
16162: SIRegister_IInternetProtocolRoot(CL);
16163: SIRegister_IInternetProtocol(CL);
16164: SIRegister_IInternetProtocolSink(CL);
16165: const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16166: SIRegister_IInternetSession(CL);
16167: SIRegister_IInternetThreadSwitch(CL);
16168: SIRegister_IInternetPriority(CL);
16169: const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16170: const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16171: const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);

```

```

16172: const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16173: const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16174: const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16175: const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16176: const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16177: const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16178: const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16179: const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16180: const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16181: const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16182: const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16183: const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16184: const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16185: const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16186: const 'PSU_DEFAULT','LongInt').SetInt( 1);
16187: const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16188: const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16189: const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16190: const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16191: const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16192: const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16193: const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16194: const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16195: const 'QUERYUSES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16196: const 'QUERY_IS_CACHED','LongInt').SetInt( QUERYUSES_NETWORK + 1);
16197: const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16198: const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16199: const 'QUERYUSES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16200: const 'QUERY_IS_SECURE','LongInt').SetInt( QUERYUSES_CACHE + 1);
16201: const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16202: SIRegister_IInternetProtocolInfo(CL);
16203: IOInet', 'IInternet');
16204: IOInetBindInfo', 'IInternetBindInfo');
16205: IOInetProtocolRoot', 'IInternetProtocolRoot');
16206: IOInetProtocol', 'IInternetProtocol');
16207: IOInetProtocolSink', 'IInternetProtocolSink');
16208: IOInetProtocolInfo', 'IInternetProtocolInfo');
16209: IOInetSession', 'IInternetSession');
16210: IOInetPriority', 'IInternetPriority');
16211: IOInetThreadSwitch', 'IInternetThreadSwitch');
16212: Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD ) : HResult';
16213: Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD ) : HResult';
16214: Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD ) : HResult';
16215: Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult';
16216: Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD ) : HResult';
16217: Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession: IInternetSes;dwReserved:DWORD):HResult;
16218: Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var pwzSecUrl:LPWSTR;psuAct:TFSUAction;dwReserv:DWORD):HResult;
16219: Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD ) : HResult';
16220: Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD ) : HResult';
16221: Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD ) : Hresult';
16222: Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD ) : HResult';
16223: Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD ) : HResult';
16224: Function OInetGetSession(dwSessionMode:DWORD; var pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16225: //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo ) : HResult';
16226: //Procedure ReleaseBindInfo( const bindinfo : TBindInfo );
16227: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ) );
16228: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ) );
16229: //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ) );
16230: //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ) );
16231: //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ) );
16232: const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16233: SIRegister_IInternetSecurityMgrSite(CL);
16234: const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16235: const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16236: const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16237: const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16238: const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16239: const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16240: const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16241: const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16242: const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16243: const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16244: const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16245: const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16246: const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16247: const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16248: const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16249: const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16250: const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16251: const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);

```

```

16252: const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16253: const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16254: const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16255: const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16256: const 'PUA_NOUIFLOCKED','LongWord').SetUInt( $00100000);
16257: const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16258: const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16259: const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16260: const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16261: SIRegister_IInternetSecurityManager(CL);
16262: SIRegister_IInternetHostSecurityManager(CL);
16263: SIRegister_IInternetSecurityManagerEx(CL);
16264: const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16265: const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16266: const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16267: const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16268: const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16269: const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16270: const 'URLACTION_ACTIVE_MIN','LongWord').SetUInt( $00001200);
16271: const 'URLACTION_ACTIVE_RUN','LongWord').SetUInt( $00001200);
16272: const 'URLACTION_ACTIVE_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16273: const 'URLACTION_ACTIVE_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16274: const 'URLACTION_ACTIVE_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16275: const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16276: const 'URLACTION_ACTIVE_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16277: const 'URLACTION_ACTIVE_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16278: const 'URLACTION_ACTIVE_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16279: const 'URLACTION_ACTIVE_CURR_MAX','LongWord').SetUInt( $00001206);
16280: const 'URLACTION_ACTIVE_MAX','LongWord').SetUInt( $000013FF);
16281: const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16282: const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16283: const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16284: const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16285: const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16286: const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16287: const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16288: const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16289: const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16290: const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16291: const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16292: const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16293: const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16294: const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16295: const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16296: const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16297: const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16298: const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16299: const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16300: const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16301: const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16302: const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16303: const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16304: const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16305: const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16306: const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16307: const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019FF);
16308: const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16309: const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16310: const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16311: const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16312: const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16313: const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16314: const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16315: const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16316: const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16317: const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16318: const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16319: const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16320: const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16321: const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16322: const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16323: const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16324: const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16325: const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16326: const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16327: const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16328: const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001cff);
16329: const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001d00);
16330: const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001d00);
16331: const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001d01);
16332: const 'URLACTION_INFODELIVERY_NO_NO_Removing_CHANNELS','LongWord').SetUInt( $00001d02);
16333: const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001d03);
16334: const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001d04);
16335: const 'URLACTION_INFODELIVERY_NO_Removing_SUBSCRIPTIONS','LongWord').SetUInt( $00001d05);
16336: const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001d06);
16337: const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001d06);
16338: const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001dff);
16339: const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001e00);
16340: const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001e05);

```

```

16341: const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT', 'LongWord').SetUInt( $00010000);
16342: const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE', 'LongWord').SetUInt( $00020000);
16343: const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL', 'LongWord').SetUInt( $00030000);
16344: const 'URLACTION_CHANNEL_SOFTDIST_MAX', 'LongWord').SetUInt( $00001EFF);
16345: const 'URLACTION_BEHAVIOR_MIN', 'LongWord').SetUInt( $00002000);
16346: const 'URLACTION_BEHAVIOR_RUN', 'LongWord').SetUInt( $00002000);
16347: const 'URLPOLICY_BEHAVIOR_CHECK_LIST', 'LongWord').SetUInt( $00010000);
16348: const 'URLACTION_FEATURE_MIN', 'LongWord').SetUInt( $00002100);
16349: const 'URLACTION_FEATURE_MIME_SNIFFING', 'LongWord').SetUInt( $00002100);
16350: const 'URLACTION_FEATURE_ZONE_ELEVATION', 'LongWord').SetUInt( $00002101);
16351: const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS', 'LongWord').SetUInt( $00002102);
16352: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN', 'LongWord').SetUInt( $00002200);
16353: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI', 'LongWord').SetUInt( $00002201);
16354: const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS', 'LongWord').SetUInt( $00002300);
16355: const 'URLPOLICY_ALLOW', 'LongWord').SetUInt( $00);
16356: const 'URLPOLICY_QUERY', 'LongWord').SetUInt( $01);
16357: const 'URLPOLICY_DISALLOW', 'LongWord').SetUInt( $03);
16358: const 'URLPOLICY_NOTIFY_ON_ALLOW', 'LongWord').SetUInt( $10);
16359: const 'URLPOLICY_NOTIFY_ON_DISALLOW', 'LongWord').SetUInt( $20);
16360: const 'URLPOLICY_LOG_ON_ALLOW', 'LongWord').SetUInt( $40);
16361: const 'URLPOLICY_LOG_ON_DISALLOW', 'LongWord').SetUInt( $80);
16362: const 'URLPOLICY_MASK_PERMISSIONS', 'LongWord').SetUInt( $0F);
16363: Function GetUrlPolicyPermissions( dw : DWORD) : DWORD';
16364: Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD';
16365: const 'URLZONE_PREDEFINED_MIN', 'LongInt').SetInt( 0);
16366: const 'URLZONE_LOCAL_MACHINE', 'LongInt').SetInt( 0);
16367: const 'URLZONE_INTRANET', 'LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16368: const 'URLZONE_TRUSTED', 'LongInt').SetInt( URLZONE_INTRANET + 1);
16369: const 'URLZONE_INTERNET', 'LongInt').SetInt( URLZONE_TRUSTED + 1);
16370: const 'URLZONE_UNTRUSTED', 'LongInt').SetInt( URLZONE_INTERNET + 1);
16371: const 'URLZONE_PREDEFINED_MAX', 'LongInt').SetInt( 999);
16372: const 'URLZONE_USER_MIN', 'LongInt').SetInt( 1000);
16373: const 'URLZONE_USER_MAX', 'LongInt').SetInt( 10000);
16374: const 'URLTEMPLATE_CUSTOM', 'LongWord').SetUInt( $00000000);
16375: const 'URLTEMPLATE_PREDEFINED_MIN', 'LongWord').SetUInt( $00010000);
16376: const 'URLTEMPLATE_LOW', 'LongWord').SetUInt( $00010000);
16377: const 'URLTEMPLATE_MEDIUM', 'LongWord').SetUInt( $00011000);
16378: const 'URLTEMPLATE_HIGH', 'LongWord').SetUInt( $00012000);
16379: const 'URLTEMPLATE_PREDEFINED_MAX', 'LongWord').SetUInt( $00020000);
16380: const 'MAX_ZONE_PATH', 'LongInt').SetInt( 260);
16381: const 'MAX_ZONE_DESCRIPTION', 'LongInt').SetInt( 200);
16382: const 'ZAFLAGS_CUSTOM_EDIT', 'LongWord').SetUInt( $00000001);
16383: const 'ZAFLAGS_ADD_SITES', 'LongWord').SetUInt( $00000002);
16384: const 'ZAFLAGS_REQUIRE_VERIFICATION', 'LongWord').SetUInt( $00000004);
16385: const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE', 'LongWord').SetUInt( $00000008);
16386: const 'ZAFLAGS_INCLUDE_INTRANET_SITES', 'LongWord').SetUInt( $00000010);
16387: const 'ZAFLAGS_NO_UI', 'LongWord').SetUInt( $00000020);
16388: const 'ZAFLAGS_SUPPORTS_VERIFICATION', 'LongWord').SetUInt( $00000040);
16389: const 'ZAFLAGS_UNC_AS_INTRANET', 'LongWord').SetUInt( $00000080);
16390: const 'ZAFLAGS_USE_LOCKED_ZONES', 'LongWord').SetUInt( $00010000);
16391: const 'ZAFLAGS_USE_LOCKED_ZONES', 'LongWord').SetUInt( $00010000);
16392: //PZoneAttributes', '^TZoneAttributes // will not work';
16393: _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');
16394: { _ZONEATTRIBUTES = packed record
16395:   cbSize: ULONG;
16396:   szDisplayName: array [0..260 - 1] of WideChar;
16397:   szDescription: array [0..200 - 1] of WideChar;
16398:   szIconPath: array [0..260 - 1] of WideChar;
16399:   dwTemplateMinLevel: DWORD;
16400:   dwTemplateRecommended: DWORD;
16401:   dwTemplateCurrentLevel: DWORD;
16402:   dwFlags: DWORD;
16403: end; }
16404: TZoneAttributes', '_ZONEATTRIBUTES');
16405: ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16406: const 'URLZONEREG_DEFAULT', 'LongInt').SetInt( 0);
16407: const 'URLZONEREG_HKLM', 'LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16408: const 'URLZONEREG_HKCU', 'LongInt').SetInt( URLZONEREG_HKLM + 1);
16409: SIRegister_IInternetZoneManager(CL);
16410: SIRegister_IInternetZoneManagerEx(CL);
16411: const 'SOFTDIST_FLAG_USAGE_EMAIL', 'LongWord').SetUInt( $00000001);
16412: const 'SOFTDIST_FLAG_USAGE_PRECACHE', 'LongWord').SetUInt( $00000002);
16413: const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL', 'LongWord').SetUInt( $00000004);
16414: const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION', 'LongWord').SetUInt( $00000008);
16415: const 'SOFTDIST_ADSTATE_NONE', 'LongWord').SetUInt( $00000000);
16416: const 'SOFTDIST_ADSTATE_AVAILABLE', 'LongWord').SetUInt( $00000001);
16417: const 'SOFTDIST_ADSTATE_DOWNLOADED', 'LongWord').SetUInt( $00000002);
16418: const 'SOFTDIST_ADSTATE_INSTALLED', 'LongWord').SetUInt( $00000003);
16419: //PCodeBaseHold', '^TCodeBaseHold // will not work';
16420: _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16421:   + 'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16422: TCodeBaseHold', '_tagCODEBASEHOLD');
16423: CODEBASEHOLD', '_tagCODEBASEHOLD');
16424: //PSoftDistInfo', '^TSoftDistInfo // will not work');
16425: _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
16426:   + 'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
16427:   + 'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'

```

```

16428:     +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
16429:     +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16430:     TSoftDistInfo', '_tagSOFTDISTINFO');
16431:     SOFTDISTINFO', '_tagSOFTDISTINFO');
16432:     SIRegister_ISoftDistExt(CL);
16433:     Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16434:     Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
dwAdvertisedVersionLS : DWORD) : HResult');
16435:     SIRegister_IDataFilter(CL);
16436: // PProtocolFilterData', '^TProtocolFilterData // will not work');
16437:     _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink :
16438:     +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16439:     +terFlags : DWORD; end');
16440:     TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16441:     PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16442: // PDataInfo', '^TDataInfo // will not work');
16443:     _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
16444:     +ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16445:     TDataInfo', '_tagDATAINFO');
16446:     DATAINFO', '_tagDATAINFO');
16447:     SIRegister_IEncodingFilterFactory(CL);
16448:     Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16449: //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL';
16450: //Function IsloggingEnabledW( pszUrl : PWideChar) : BOOL';
16451: //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16452:     _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16453:     +rlName : LPSTR; StartTime : TSystemTime; EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16454:     THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16455:     HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16456:     Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16457: end;
16458:
16459: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16460: begin
16461:     Procedure reformatMemo( const m : TCustomMemo)');
16462:     Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16463:     Procedure MoveToTop( memo : TMemo)');
16464:     Procedure ScrollToTop( memo : TMemo)');
16465:     Function LineNumberClicked( memo : TMemo) : integer');
16466:     Function MemoClickedLine( memo : TMemo) : integer');
16467:     Function ClickedMemoLine( memo : TMemo) : integer');
16468:     Function MemoLineClicked( memo : TMemo) : integer');
16469:     Function LinePositionClicked( Memo : TMemo) : integer');
16470:     Function ClickedMemoPosition( memo : TMemo) : integer');
16471:     Function MemoPositionClicked( memo : TMemo) : integer');
16472:     Procedure AdjustGridSize( grid : TDrawGrid)');
16473:     Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16474:     Procedure InsertGridRow( Grid : TStringGrid; const ARow : integer)');
16475:     Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer)');
16476:     Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean)');
16477:     Procedure sortstrDown( var s : string)');
16478:     Procedure sortstrUp( var s : string)');
16479:     Procedure rotatestrleft( var s : string)');
16480:     Function dffstrtofloatdef( s : string; default : extended) : extended');
16481:     Function deblank( s : string) : string');
16482:     Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16483:     Procedure FreeAndClearListBox( C : TListBox)');
16484:     Procedure FreeAndClearMemo( C : TMemo)');
16485:     Procedure FreeAndClearStringList( C : TStringList)');
16486:     Function dffgetfilesize( f : TSearchrec) : int64');
16487: end;
16488:
16489: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16490: begin
16491:     CL.AddTypeS('intset', 'set of byte');
16492:     TPoint64', 'record x : int64; y : int64; end');
16493:     Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16494:     Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16495:     Function GeneratePentagon( n : integer) : integer');
16496:     Function IsPentagon( p : integer) : boolean');
16497:     Function isSquare( const N : int64) : boolean');
16498:     Function isCube( const N : int64) : boolean');
16499:     Function isPalindrome( const n : int64) : boolean');
16500:     Function isPalindromel( const n : int64; var len : integer) : boolean');
16501:     Function GetEulerPhi( n : int64) : int64');
16502:     Function dffIntPower( a, b : int64) : int64');
16503:     Function IntPowerl( a : extended; b : int64) : extended');
16504:     Function gcd2( a, b : int64) : int64');
16505:     Function GCDMany( A : array of integer) : integer');
16506:     Function LCMMany( A : array of integer) : integer');
16507:     Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16508:     Function dffFactorial( n : int64) : int64');
16509:     Function digitcount( n : int64) : integer');
16510:     Function nextpermute( var a : array of integer) : boolean');
16511:     Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16512:     Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16513:     Function InttoBinaryStr( nn : integer) : string');
16514:     Function StrtoAngle( const s : string; var angle : extended) : boolean');
16515:     Function AngleToStr( angle : extended) : string');

```

```

16516: Function deg2rad( deg : extended ) : extended');
16517: Function rad2deg( rad : extended ) : extended');
16518: Function GetLongToMercProjection( const long : extended ) : extended');
16519: Function GetLatToMercProjection( const Lat : Extended ) : Extended');
16520: Function GetMercProjectionToLong( const ProjLong : extended ) : extended');
16521: Function GetMercProjectionToLat( const ProjLat : extended ) : extended');
16522: SIRegister_TPrimes(CL);
16523: //RIRegister_TPrimes(CL);
16524: //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16525: CL.AddConstantN('minmark','LongInt').SetInt( 180);
16526: Function Random64( const N : Int64 ) : Int64;');
16527: Procedure Randomize64());
16528: Function Random641 : extended');
16529: Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist )//DFFUUtils
16530: end;
16531:
16532: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16533: begin
16534:   TrealPoint', 'record x : extended; y : extended; end');
16535:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16536:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16537:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16538:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16539:   PPResult', '( PPointSide, PPIInside, PPVertex, PPEdge, PPError )');
16540:   Function realpoint( x, y : extended ) : TRealPoint');
16541:   Function dist( const p1, p2 : TrealPoint ) : extended');
16542:   Function intdist( const p1, p2 : TPoint ) : integer');
16543:   Function dffLine( const p1, p2 : TPoint ) : Tline');
16544:   Function Line1( const p1, p2 : TRealPoint ) : TRealLine');
16545:   Function dffCircle( const cx, cy, R : integer ) : TCircle');
16546:   Function Circle1( const cx, cy, R : extended ) : TRealCircle');
16547:   Function GetTheta( const L : TLine ) : extended');
16548:   Function GetTheta1( const p1, p2 : TPoint ) : extended');
16549:   Function GetTheta2( const p1, p2 : TRealPoint ) : extended');
16550:   Procedure Extendline( var L : TLine; dist : integer );
16551:   Procedure Extendline1( var L : TRealLine; dist : extended );
16552:   Function Linesintersect( line1, line2 : TLine ) : boolean');
16553:   Function ExtendedLinesIntersect( Line1, Line2 : TLine; const extendlines : bool; var IP : TPoint ) : bool;
16554:   Function ExtendedLinesIntersect1( const Line1, Line2 : TLine; const extendlines : bool; var IP : TRealPoint ) : bool;
16555:   Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint ) : boolean';
16556:   Function PointPerpendiculararline( L : TLine; P : TPoint ) : TLine );
16557:   Function PerpDistance( L : TLine; P : TPoint ) : Integer );
16558:   Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended ) : TLine );
16559:   Function AngledLineFromLine1( L : TLine; P : TPoint; Dist : extended; alpha : extended; useScreenCoordinates : bool ) : TLine;
16560:   Function PointInPoly( const p : TPoint; Points : array of TPoint ) : PPResult );
16561:   Function PolygonArea( const points : array of TPoint; const screencoordinates : boolean; var Clockwise : bool );
16562:   Procedure InflatePolygon( const points : array of TPoint; var points2 : array of TPoint; var area : integer; const screenCoordinates : boolean; const inflateby : integer );
16563:   Function PolyBuiltClockwise( const points : array of TPoint; const screencoordinates : boolean ) : bool;
16564:   Function DegtosRad( d : extended ) : extended );
16565:   Function RadtoDeg( r : extended ) : extended );
16566:   Procedure TranslateLeftTo( var L : TLine; newend : TPoint );
16567:   Procedure TranslateLeftTol( var L : TrealLine; newend : TrealPoint );
16568:   Procedure RotateRightEndBy( var L : TLine; alpha : extended );
16569:   Procedure RotateRightEndTo( var L : TLine; alpha : extended );
16570:   Procedure RotateRightEndTol( var p1, p2 : Trealpoint; alpha : extended );
16571:   Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, IP2 : TPoint ) : boolean );
16572:   Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, IP2 : TRealPoint ) : boolean );
16573:   Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine ) : boolean );
16574:   Function CircleCircleExtTangentLines( C1, C2 : TCircle; var C3 : TCircle; var L1, L2, PL1, PL2, TL1, TL2 : TLine ) : Bool;
16575: end;
16576:
16577:
16578: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16579: begin
16580:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGallLonLat )');
16581:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16582:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16583:   TSunrec', 'record TrueEclLon : extended;
16584:     AppEclLon : extended; AUDistance : extended; TrueHADecl : TRPoint; TrueRADecl : TRPoint;
16585:     TrueAzAlt : TRPoint; AppHADecl : TRPoint; AppRADecl : TRPoint; AppAzAlt : TRPoint; SunMeanAnomaly : extended; end;
16586:     TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl : '
16587:       + ' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16588:       + 'arth : extended; Phase : extended; end');
16589:   TMDatetime', 'record UmbralStartTime : TDatetime; UmbralEnd'
16590:     + 'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDatetime; end');
16591:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16592:     + 'ct : TDatetime; LastContact : Date; Magnitude : Extended; MaxEclipseUTime : Date; end');
16593:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16594:   TPlanetRec', 'record AsOf : Date; Name : string; MeanLon : '
16595:     + 'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16596:     + 'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16597:     + 'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16598:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRPoint; RadiusVector : '
16599:     extended; UncorrectedEarthDistance : extended; GeoEclLonLat : TRPoint; CorrectedEarthDistance : extended;
16600:     ApparentRaDecl : TRPoint; end');
16601:   SIRegister_TAstronomy(CL);

```

```

16598: Function AngleToStr( angle : extended ) : string');
16599: Function StrToAngle( s : string; var angle : extended ) : boolean');
16600: Function HoursToStr24( t : extended ) : string');
16601: Function RPoint( x, y : extended ) : TRPoint');
16602: Function getStimename( t : TDTType ) : string');
16603: end;
16604:
16605: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16606: begin
16607:   TCardValue', 'Integer');
16608:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts ');
16609:   TShortSuit', '( cardS, cardD, cardC, cardH ');
16610:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16611:   SIRegister_TCard(CL);
16612:   SIRegister_TDeck(CL);
16613: end;
16614:
16615: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16616: begin
16617:   tMethodCall', 'Procedure');
16618:   tVerboseCall', 'Procedure ( s : string)');
16619: // PTEdge', '^TEdge // will not work');
16620:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer;
16621:     +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16622:   SIRegister_TNode(CL);
16623:   SIRegister_TGraphList(CL);
16624: end;
16625:
16626: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16627: begin
16628:   ParserFloat', 'extended');
16629: //PParserFloat', '^ParserFloat // will not work');
16630:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16631:     +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar ');
16632: //POperation', '^TOperation // will not work');
16633:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16634:     +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16635:     +'; Token : TDFFToken; end');
16636: TMathProcedure', 'procedure(AnOperation: TDFFOperation)');
16637: (CL.FindClass('TOBJECT'),'EMathParserError');
16638: CL.FindClass('TOBJECT'),'ESyntaxError');
16639: (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16640: (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16641: (CL.FindClass('TOBJECT'),'ETooManyNestings');
16642: (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16643: (CL.FindClass('TOBJECT'),'EBadName');
16644: (CL.FindClass('TOBJECT'),'EParserInternalError');
16645: ('TEXParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16646: SIRegister_TCustomParser(CL);
16647: SIRegister_TExParser(CL);
16648: end;
16649:
16650: function isService: boolean;
16651: begin
16652:   result:= NOT(Application is TApplication);
16653:   {result:= Application is TServiceApplication;}
16654: end;
16655: function isApplication: boolean;
16656: begin
16657:   result:= Application is TApplication;
16658: end;
16659: //SM_REMOTESESSION = $1000
16660: function isTerminalSession: boolean;
16661: begin
16662:   result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16663: end;
16664:
16665: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16666: begin
16667:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header :
16668:   +'String; margin_bottom : String; margin_left : String; margin_right : String';
16669:   +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16670:   Function cyURLEncode( const S : string ) : string';
16671:   Function MakeResourceURL( const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16672:   Function MakeResourceURL1( const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;
16673:   Function cyColorToHtml( aColor : TColor ) : String');
16674:   Function HtmlToColor( aHtmlColor : String ) : TColor');
16675: //Function GetStreamEncoding( aStream : TStream ) : TEncoding';
16676: // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding ) : Boolean';
16677: Function AddHtmlUnicodePrefix( aHtml : String ) : String');
16678: Function RemoveHtmlUnicodePrefix( aHtml : String ) : String');
16679: Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup );
16680: Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup );
16681: CL.AddConstantN('IEBodyBorderless','String').SetString('none');
16682: CL.AddConstantN('IEBodySingleBorder','String').SetString('');
16683: CL.AddConstantN('IEDesignModeOn','String').SetString('On');
16684: CL.AddConstantN('IEDesignModeOff','String').SetString('Off');
16685: CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF );
16686: CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE );

```

```

16687: end;
16688:
16689:
16690: procedure SIRegister_UcomboV2(CL: TPSPPascalCompiler);
16691: begin
16692:   CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600 );
16693:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16694:     +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16695:     +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16696:     +'inationsRepeat, CombinationsRepeatDown )');
16697:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16698:   SIRegister_TComboSet(CL);
16699: end;
16700:
16701: procedure SIRegister_cyBaseComm(CL: TPSPPascalCompiler);
16702: begin
16703:   TcyCommandType', '( ctDeveloperDefined, ctUserDefined )');
16704:   TStreamContentType', '( scDeveloperDefined, scUserDefined, scString )');
16705:   TProcOnReceiveCommand', 'Procedure ( Sender: TObject; aCommand: Word; userParam : Integer )';
16706:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16707:     +'mmHandle : THandle; aString : String; userParam : Integer )';
16708:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16709:     +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer )';
16710:   SIRegister_TcyBaseComm(CL);
16711:   CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1 );
16712:   CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99 );
16713:   Function ValidatefileMappingName( aName : String ) : String );
16714:   procedure makeCaption(leftSide, Rightside:string; form:TForm);
16715: end;
16716:
16717: procedure SIRegister_cyDERUtils(CL: TPSPPascalCompiler);
16718: begin
16719:   CL.AddTypeS('DERString', 'String');
16720:   CL.AddTypeS('DERChar', 'Char');
16721:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16722:     +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16723:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
16724:   CL.AddTypeS('DERNString', 'String');
16725:   const DERDecimalSeparator', 'String').SetString( '.' );
16726:   const DERDefaultChars', 'String')('+@/%-'
 $\cdot 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ')$ 
16727:   const DERDefaultChars', 'String').SetString( '/%-0123456789abcdefghijklmnopqrstuvwxyz' );
16728:   CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char ) : Boolean');
16729:   Function isValidWebMailChar( aChar : Char ) : Boolean';
16730:   Function isValidwebSite( aStr : String ) : Boolean';
16731:   Function isValidWebMail( aStr : String ) : Boolean';
16732:   Function ValidateDate( aDERStr : DERString; var RsltFormat : String ) : Boolean';
16733:   Function DERStrToDate( aDERStr, aFormat : String ) : TDate';
16734:   Function IsDERChar( aChar : Char ) : Boolean';
16735:   Function IsDERDefaultChar( aChar : Char ) : Boolean';
16736:   Function IsDERMoneyChar( aChar : Char ) : Boolean';
16737:   Function IsDERExceptionChar( aChar : Char ) : Boolean';
16738:   Function IsDERSymbols( aDERString : String ) : Boolean';
16739:   Function StringToDERCharSet( aStr : String ) : DERString';
16740:   Function IsDERNDefaultChar( aChar : Char ) : Boolean';
16741:   Function IsDERNChar( aChar : Char ) : Boolean';
16742:   Function DERToDERNCharset( aDERStr : DERString ) : DERNString';
16743:   Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean ) : String';
16744:   Function DERExtractWebMail( aDERStr : DERString ) : String';
16745:   Function DERExtractPhoneNr( aDERStr : DERString ) : String';
16746:   Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
SmartWebsiteRecognition:Boolean):TElementsType;
16747:   Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
SmartWebsiteRecognition:Boolean):TElementsType;
16748:   Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
aElementsType : TElementsType ) : String';
16749:   Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType );';
16750: end;
16751:
16752: procedure SIRegister_cyImage(CL: TPSPPascalCompiler);
16753: begin
16754:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16755:   Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer';
16756:   Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean );
16757:   Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean );
16758:   Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean );
16759:   Procedure BitmapSetPixelsContrast( Bmp : TBitmap; Incpixels : Integer; RefreshBmp : Boolean );
16760:   Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean );
16761:   Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16762:   Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16763:   Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16764:   Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
SingleDestinationColor, RefreshBmp : Boolean);';
16765:   Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean );

```

```

16766: Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor )';
16767: Procedure BitmapBlur( SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent : Integer;RefreshBmp:Bool;
16768: Procedure GraphicMirror( Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16769: Procedure GraphicMirror1( Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean );');
16770: Function BitmapCreate( BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat ):TBitmap;
16771: Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word );
16772: Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String );
16773: end;
16774:
16775: procedure SIRegister_WaveUtils(CL: TPSPPascalCompiler);
16776: begin
16777:   TMS2StrFormat', '( msHMS, msHMS, msMS, msSh, msS, msAh,msA )');
16778:   TPCMChannel', '( cMono, cStereo )');
16779:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16780:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16781:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono16b';
16782:   +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b';
16783:   +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b';
16784:   +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b';
16785:   +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b';
16786:   +'it48000Hz, Stereo16bit48000Hz )');
16787: PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16788:   +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word; ecbSiz: Word; end');
16789: tWaveFormatEx', 'PWaveFormatEx');
16790: HMMIO', 'Integer');
16791: TWaveDeviceFormats', 'set of TPCMFormat');
16792: TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16793:   +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16794: CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16795: TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16796: TWaveOutOptions', 'set of TWaveOutOption');
16797: TStreamOwnership2', '( soReference, soOwned )');
16798: TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16799: // PRawWave', '^TRawWave // will not work');
16800: TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16801: CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioError');
16802: CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioSysError');
16803: CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioInvalidOperation');
16804: TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16805: TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW';
16806:   +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16807: TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf';
16808:   +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16809: TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu';
16810:   +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16811: TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const ';
16812:   +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16813: TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16814: TWaveAudioFilterEvent', 'Procedure ( Sender : TObject; const Buffer:TObject; BufferSize:DWORD)');
16815: GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16816: CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16817: CloseWaveAudio(mmIO: HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16818: GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16819: CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16820: OpenStreamWaveAudio( Stream : TStream) : HMMIO');
16821: CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16822: GetAudioFormat( FormatTag : Word ) : String');
16823: GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx ) : String');
16824: GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD ) : DWORD');
16825: GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx ) : DWORD');
16826: GetWaveAudioPeakLevel( const Data: TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16827: InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx: Boolean');
16828: SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx: Boolean');
16829: ChangeWaveAudioVolume( const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16830: MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
16831: TObject; BufferSize : DWORD) : Boolean');
16832: ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16833: SetPCMFormat( const pWaveFormat : PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec : TPCMSamplesPerSec; BitsPerSample : TPCMbitsPerSample );
16834: Procedure SetPCMAudioFormats( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat );
16835: GetPCMFormat( const pWaveFormat : PWaveFormatEx ) : TPCMFormat');
16836: GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD ) : DWORD');
16837: MS2str( Milliseconds : DWORD; Fmt : TMS2StrFormat ) : String');
16838: WaitForSyncObject( SyncObject : THandle; Timeout : DWORD ) : DWORD');
16839: //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD ) : LRESULT');
16840:
16841: procedure SIRegister_NamedPipes(CL: TPSPPascalCompiler);
16842: begin
16843:   'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096 );
16844:   CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000 );
16845:   'PIPE_NAMING_SCHEME','String').SetString( '\%s\pipe\%s' );
16846:   'WAIT_ERROR','LongWord').SetUInt( DWORD( $FFFFFF ) );
16847:   'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1 );
16848:   'STATUS_SUCCESS','LongWord').SetUInt( $00000000 );
16849:   'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005 );
16850:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16851:   CL.AddTypeS('TPipeType', '( ptyp_BytByte, ptyp_MsgByte, ptyp_MsgMsg )');

```

```

16852: CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16853: SIRegister_TNamedPipe(CL);
16854: SIRegister_TSserverPipe(CL);
16855: SIRegister_TClientPipe(CL);
16856: CL.AddDelphiFunction('Function CalculateTimeout( aBasis : DWORD ) : DWORD');
16857: Function HasOverlappedIoCompleted( const ov : OVERLAPPED ) : Boolean';
16858: Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean): OverlappedResult;
16859: Function GetStreamAsText( stm : TStream ) : string';
16860: Procedure SetStreamAsText( const aTxt : string; stm : TStream )';
16861: end;
16862:
16863: procedure SIRegister_DPUtils(CL: TPSPascalCompiler);
16864: begin
16865: // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16866: SIRegister_TThumbData(CL);
16867: 'PIC_BMP','LongInt').SetInt( 0 );
16868: 'PIC_JPG','LongInt').SetInt( 1 );
16869: 'THUMB_WIDTH','LongInt').SetInt( 60 );
16870: 'THUMB_HEIGHT','LongInt').SetInt( 60 );
16871: Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime) : Boolean';
16872: Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)';
16873: Function ReadBitmap( Filehandle : Integer; Width, Height : Integer ) : TBitmap';
16874: Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap )';
16875: Function OpenPicture( fn : string; var tp : Integer ) : Integer';
16876: Function ConvertPicture( pi : Integer; tp : Integer ) : TBitmap';
16877: Function LoadPicture( fn : string; var w, h : Integer ) : TBitmap';
16878: Function TurnBitmap( bmp : TBitmap; ang : Integer ) : TBitmap';
16879: Function RotateBitmap( Bitmap : TBitmap; Direction : Integer ) : TBitmap';
16880: Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap ) : TRect';
16881: Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer ) : TBitmap';
16882: Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer )';
16883: Procedure FindFiles( path, mask : string; items : TStringList )';
16884: Function LetFileName( s : string ) : string';
16885: Function LetParentPath( path : string ) : string';
16886: Function AddBackSlash( path : string ) : string';
16887: Function CutBackSlash( path : string ) : string';
16888: end;
16889:
16890: procedure SIRegister_CommonTools(CL: TPSPascalCompiler);
16891: begin
16892: //BYTES','LongInt').SetInt( 1 );
16893: 'KBYTES ','LongInt').SetInt( 1024 );
16894: 'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ) );
16895: 'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ) );
16896: 'DBG_GONE','LongWord').SetUInt( $99AC1D99 );
16897: 'SHELL_NS_MYCOMPUTER','String').SetString( ':{20D04FE0-3AEA-1069-A2D8-08002B30309D}' );
16898: SIRegister_MakeComServerMethodsPublic(CL);
16899: CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16900: Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD ) : Boolean';
16901: Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean )';
16902: Function TBGetTempFolder : string';
16903: Function TBGetTempFile : string';
16904: Function TBGetModuleFilename : string';
16905: Function FormatModuleVersionInfo( const aFilename : string ) : string';
16906: Function GetVersionInfoString( const afile, aEntry : string; aLang : WORD ) : string';
16907: Function TBGetFileSize( aFile : string; aMultipleOf : Integer ) : Integer';
16908: Function FormatAttribString( aAttr : Integer ) : string';
16909: Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo ) : string';
16910: Function ShellRecycle( aWnd : HWND; aFileOrFolder : string ) : Boolean';
16911: Function IsDebuggerPresent : BOOL';
16912: Function TBNotImplemented : HRESULT';
16913: end;
16914:
16915: procedure SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
16916: begin
16917: //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16918: //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16919: CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16920: CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean');
16921: //TDrivesProperty = array['A'..'Z'] of boolean;
16922: Function TBSetSystemTime( DateTime : TDateTime; DOW : word ) : boolean';
16923: Function IsElevated : Boolean';
16924: Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:GUID;const aIID:GUID;out aObj:TObject);
16925: CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu Defined )');
16926: Function TrimNetResource( UNC : string ) : string';
16927: Procedure GetFreeDrives( var FreeDrives : TDrivesProperty );
16928: Procedure GetMappedDrives( var MappedDrives : TDrivesProperty );
16929: Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
UserUsage : TPasswordUsage; User : string; Comment : string ) : boolean';
16930: Function UnmapDrive( Drive : char; Force : boolean ) : boolean';
16931: Function TBIsWindowsVista : Boolean';
16932: Procedure SetVistaFonts( const AForm : TForm )';
16933: Procedure SetVistaContentFonts( const AFont : TFont )';
16934: Function GetProductType( var sType : String ) : Boolean';
16935: Function lstrcmp( lpString1, lpString2 : PChar ) : Integer';
16936: Function lstrcmpi( lpString1, lpString2 : PChar ) : Integer';
16937: Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer ) : PChar';
16938: Function lstrcpy( lpString1, lpString2 : PChar ) : PChar ';

```

```

16939: Function lstrcat( lpString1, lpString2 : PChar ) : PChar');
16940: Function lstrlen( lpString : PChar ) : Integer');
16941: Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTTokenInformationClass;
TokenInformation : __Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16942: Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTTokenInformationClass;
TokenInformation : __Pointer; TokenInformationLength : DWORD) : BOOL');
16943: end;
16944:
16945: procedure SIRегистre_dwsXPlatform(CL: TPSPascalCompiler);
16946: begin
16947:   'cLineTerminator', 'Char').SetString( #10);
16948:   'cLineTerminators', 'String').SetString( #13#10);
16949:   'INVALID_HANDLE_VALUE', 'LongInt').SetInt( DWORD( - 1));
16950:   SIRегистre_TFixedCriticalSection(CL);
16951:   SIRегистre_TMultiReadSingleWrite(CL);
16952:   CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16953:   Function GetDecimalSeparator : Char');
16954:   TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16955:   Procedure Collectfiles(const directory,fileMask:UnicodeString; list:TStrings;
recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16956:   CL.AddTypeS('NativeInt', 'Integer');
16957:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16958:   CL.AddTypeS('NativeUInt', 'Cardinal');
16959:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16960:   //CL.AddTypeS('TBytes', 'array of Byte');
16961:   CL.AddTypeS('RawByteString', 'UnicodeString');
16962:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16963:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16964:   SIRегистre_TPath(CL);
16965:   SIRегистre_TFile(CL);
16966:   SIRегистre_TdwsThread(CL);
16967:   Function GetSystemMilliseconds : Int64';
16968:   Function UTCDateTime : TDateTime');
16969:   Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString';
16970:   Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer';
16971:   Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer';
16972:   Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer';
16973:   Function UnicodeComparePChars( pl : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer';
16974:   Function UnicodeComparePChars1( pl, p2 : PChar; n : Integer) : Integer';
16975:   Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString';
16976:   Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString';
16977:   Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer';
16978:   Function ASCIIISameText( const s1, s2 : UnicodeString) : Boolean';
16979:   Function InterlockedIncrement( var val : Integer) : Integer';
16980:   Function InterlockedDecrement( var val : Integer) : Integer';
16981:   Procedure FastInterlockedIncrement( var val : Integer)';
16982:   Procedure FastInterlockedDecrement( var val : Integer)';
16983:   Function InterlockedExchangePointer( var target : __Pointer; val : __Pointer) : __Pointer';
16984:   Procedure SetThreadName( const threadName : Char; threadID : Cardinal)';
16985:   Procedure dwsOutputDebugString( const msg : UnicodeString)';
16986:   Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16987:   Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16988:   Procedure VarCopy( out dest : Variant; const src : Variant)';
16989:   Function VarToUnicodeStr( const v : Variant) : UnicodeString';
16990:   Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString';
16991:   Function LoadTextFromStream( aStream : TStream) : UnicodeString';
16992:   Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString';
16993:   Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)';
16994:   Function OpenFileForSequentialReadonly( const fileName : UnicodeString) : THandle';
16995:   Function OpenFileForSequentialWriteonly( const fileName : UnicodeString) : THandle';
16996:   Procedure CloseFileHandle( hFile : THandle)';
16997:   Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean';
16998:   Function FileMove( const existing, new : UnicodeString) : Boolean';
16999:   Function dwsfileDelete( const fileName : String) : Boolean';
17000:   Function FileRename( const oldName, newName : String) : Boolean';
17001:   Function dwsfileSize( const name : String) : Int64';
17002:   Function dwsfileDateTime( const name : String) : TDateTime';
17003:   Function DirectSet8087CW( newValue : Word) : Word';
17004:   Function DirectSetMXCSR( newValue : Word) : Word';
17005:   Function TtoObject( const T: byte) : TObject';
17006:   Function TtoPointer( const T: byte) : __Pointer';
17007:   Procedure GetMemForT(var T: byte; Size : integer)';
17008:   Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer';
17009: end;
17010:
17011: procedure SIRегистre_AdSocket(CL: TPSPascalCompiler);
17012: begin
17013:   'IPStrSize', 'LongInt').SetInt( 15);
17014:   'CM_APDSOCKETMESSAGE', 'LongWord').SetUInt( WM_USER + $0711);
17015:   'CM_APDSOCKETQUIT', 'LongWord').SetUInt( WM_USER + $0712);
17016:   'ADWSBASE', 'LongInt').SetInt( 9000);
17017:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
+ 'SelectEvent : Word; SelectError : Word; Result : Longint; end');
17018:   SIRегистre_EApdSocketException(CL);
17020:   TWsMode', '( wsClient, wsServer )');
17021:   TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket)';
17022:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrorCode : Integer)');
17023:   SIRегистre_TApdSocket(CL);
17024: end;

```

```

17025:
17026: procedure SIRегистер_AdPort(CL: TPSCompiler);
17027: begin
17028:   SIRегистер_TApdCustomComPort(CL);
17029:   SIRегистер_TApdComPort(CL);
17030:   Function ComName( const ComNumber : Word ) : ShortString';
17031:   Function SearchComPort( const C : TComponent ) : TApdCustomComPort';
17032: end;
17033:
17034: procedure SIRегистер_PathFunc(CL: TPSCompiler);
17035: begin
17036:   Function inAddBackslash( const S : String ) : String';
17037:   Function PathChangeExt( const Filename, Extension : String ) : String';
17038:   Function PathCharCompare( const S1, S2 : PChar ) : Boolean';
17039:   Function PathCharIsSlash( const C : Char ) : Boolean';
17040:   Function PathCharIsTrailByte( const S : String; const Index : Integer ) : Boolean';
17041:   Function PathCharLength( const S : String; const Index : Integer ) : Integer';
17042:   Function inPathCombine( const Dir, Filename : String ) : String';
17043:   Function PathCompare( const S1, S2 : String ) : Integer';
17044:   Function PathDrivePartLength( const Filename : String ) : Integer';
17045:   Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;
17046:   Function inPathExpand( const Filename : String ) : String';
17047:   Function PathExtensionPos( const Filename : String ) : Integer';
17048:   Function PathExtractDir( const Filename : String ) : String';
17049:   Function PathExtractDrive( const Filename : String ) : String';
17050:   Function PathExtractExt( const Filename : String ) : String';
17051:   Function PathExtractName( const Filename : String ) : String';
17052:   Function PathExtractPath( const Filename : String ) : String';
17053:   Function PathIsRooted( const Filename : String ) : Boolean';
17054:   Function PathLastChar( const S : String ) : PChar';
17055:   Function PathLastDelimiter( const Delimiters, S : string ) : Integer';
17056:   Function PathLowercase( const S : String ) : String';
17057:   Function PathNormalizeSlashes( const S : String ) : String';
17058:   Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
17059:   Function PathPos( Ch : Char; const S : String ) : Integer';
17060:   Function PathStartsWith( const S, AStartsWith : String ) : Boolean';
17061:   Function PathStrNextChar( const S : PChar ) : PChar';
17062:   Function PathStrPrevChar( const Start, Current : PChar ) : PChar';
17063:   Function PathStrScan( const S : PChar; const C : Char ) : PChar';
17064:   Function inRemoveBackslash( const S : String ) : String';
17065:   Function RemoveBackslashUnlessRoot( const S : String ) : String';
17066:   Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean );
17067: end;
17068:
17069:
17070: procedure SIRегистер_CmnFunc2(CL: TPSCompiler);
17071: begin
17072:   NEWREGSTR_PATH_SETUP', 'String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17073:   NEWREGSTR_PATH_EXPLORER', 'String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17074:   NEWREGSTR_PATH_SPECIAL_FOLDERS', 'String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17075:   NEWREGSTR_PATH_UNINSTALL', 'String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17076:   NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME', 'String').SetString( 'DisplayName');
17077:   NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE', 'String').SetString( 'UninstallString');
17078:   KEY_WOW64_64KEY', 'LongWord').SetUInt( $0100);
17079:   //CL.AddTypeS('PLeadByteSet', '^TLeadByteSet // will not work');
17080:   CL.AddTypeS('TLeadByteSet', 'set of Char');
17081:   SIRегистер_TOneShotTimer(CL);
17082:   CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17083:   'RegViews64Bit', 'LongInt').Value.ts32 := ord(rv64Bit);
17084:   Function NewfileExists( const Name : String ) : Boolean';
17085:   Function inDirExists( const Name : String ) : Boolean';
17086:   Function FileOrDirExists( const Name : String ) : Boolean';
17087:   Function IsDirectoryAndNotReparsePoint( const Name : String ) : Boolean';
17088:   Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String';
17089:   Function GetInitInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17090:   Function GetInitBool( const Section,Key:String; const Default:Boolean;const Filename:String): Boolean';
17091:   Function IniKeyExists( const Section, Key, Filename : String ) : Boolean';
17092:   Function IsInSectionEmpty( const Section, Filename : String ) : Boolean';
17093:   Function SetIniString( const Section, Key, Value, Filename : String ) : Boolean';
17094:   Function SetInitInt(const Section,Key:String;const Value: Longint;const Filename: String):Boolean';
17095:   Function SetInitBool(const Section,Key:String;const Value: Boolean;const Filename: String):Boolean';
17096:   Procedure DeleteIniEntry( const Section, Key, Filename : String );
17097:   Procedure DeleteIniSection( const Section, Filename : String );
17098:   Function GetEnv( const EnvVar : String ) : String';
17099:   Function GetCmdTail : String';
17100:   Function GetCmdTailEx( StartIndex : Integer ) : String';
17101:   Function NewParamCount : Integer';
17102:   Function NewParamStr( Index : Integer ) : string';
17103:   Function AddQuotes( const S : String ) : String';
17104:   Function RemoveQuotes( const S : String ) : String';
17105:   Function inGetShortName( const LongName : String ) : String';
17106:   Function inGetWinDir : String';
17107:   Function inGetSystemDir : String';
17108:   Function GetSysWow64Dir : String';
17109:   Function GetSysNativeDir( const IsWin64 : Boolean ) : String';
17110:   Function inGetTempDir : String';
17111:   Function StringChange( var S : String; const FromStr,ToStr : String ) : Integer';
17112:   Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17113:   Function AdjustLength( var S : String; const Res : Cardinal ) : Boolean';

```

```

17114: Function UsingWinNT : Boolean');
17115: Function ConvertConstPercentStr( var S : String) : Boolean');
17116: Function ConvertPercentStr( var S : String) : Boolean');
17117: Function ConstPos( const Ch : Char; const S : String) : Integer');
17118: Function SkipPastConst( const S : String; const Start : Integer) : Integer');
17119: Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String) : Boolean');
17120: Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String):Boolean;
17121: Function RegValueExists( H : HKEY; Name : PChar) : Boolean');
17122: Function Reg.ReadKeyExView(const RegView:TRegView;hKey:HKEY;lpSubKey:PChar;Reserved:DWORD;lpClass:PChar;
dwOptions:DWORD;samDesired:REGSAM;lpSecurityAttributes:TObject;var phkResult:HKEY;lpdwDisposition:DWORD):Longint;
phkResult:HKEY;lpdwDisposition:DWORD):Longint;
17123: Function RegOpenKeyExView(const RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17124: Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar) : Longint;
17125: Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17126: Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyName:PChar):Longint;
17127: Function GetShellFolderPath( const FolderID : Integer ) : String');
17128: Function IsAdminLoggedOn : Boolean');
17129: Function IsPowerUserLoggedOn : Boolean');
17130: Function IsMultiByteString( const S : AnsiString) : Boolean');
17131: Function FontExists( const FaceName : String) : Boolean');
17132: //Procedure FreeAndNil( var Obj)');
17133: Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT) : HMODULE');
17134: Function GetUILanguage : LANGID');
17135: Function RemoveAccelChar( const S : String) : String');
17136: Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17137: Function AddPeriod( const S : String) : String');
17138: Function GetExceptMessage : String');
17139: Function GetPreferredUIFont : String');
17140: Function IsWildcard( const Pattern : String) : Boolean');
17141: Function WildcardMatch( const Text, Pattern : PChar) : Boolean');
17142: Function IntMax( const A, B : Integer) : Integer');
17143: Function Win32ErrorString( ErrorCode : Integer) : String');
17144: Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet)');
17145: Function inCompareMem( P1, P2 : TObject; Length : Integer) : Boolean');
17146: Function DeleteDirTree( const Dir : String) : Boolean');
17147: Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
17148: Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT)');
17149: // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17150: Function inCharInSet( C : Char; const CharSet : TSysCharSet) : Boolean');
17151: Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String) : Boolean');
17152: Function ShutdownBlockReasonDestroy( Wnd : HWND) : Boolean');
17153: Function TryStrToBoolean( const S : String; var BoolResult : Boolean) : Boolean');
17154: Procedure WaitMessageWithTimeout( const Milliseconds : DWORD)');
17155: Function MoveFileReplace(const ExistingFileName, NewFileName : String) : Boolean');
17156: Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND)');
17157: end;
17158:
17159: procedure SIRegister_CmnFunc(CL: TPPascalCompiler);
17160: begin
17161:   SIRegister_TWindowDisabler(CL);
17162:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17163:   TMsgBoxCallbackFunc', 'procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17164:   Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox)');
17165:   Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17166:   Function AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Integer');
17167:   Function MsgBoxP( const Text,Caption: PChar; const Typ: TMMsgBoxType;const Buttons:Cardinal):Int;
17168:   Function inMsgBox(const Text,Caption:String; const Typ: TMMsgBoxType;const Buttons:Cardinal):Int;
17169:   Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
Typ:TMMsgBoxType;const Buttons:Cardinal):Integer');
17170:   Procedure ReactivateTopWindow');
17171:   Procedure SetMessageBoxCaption( const Typ : TMMsgBoxType; const NewCaption : PChar)');
17172:   Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean)');
17173:   Procedure SetMessageBoxCallbackFunc(const AFunc : TMMsgBoxCallbackFunc; const AParam : LongInt)');
17174: end;
17175:
17176: procedure SIRegister_ImageGrabber(CL: TPPascalCompiler);
17177: begin
17178:   SIRegister_TImageGrabber(CL);
17179:   SIRegister_TCaptureDrivers(CL);
17180:   SIRegister_TCaptureDriver(CL);
17181: end;
17182:
17183: procedure SIRegister_SecurityFunc(CL: TPPascalCompiler);
17184: begin
17185:   Function GrantPermissionOnFile(const DisableFsRedir:Boolean; FIlename:String;const
Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean');
17186:   Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean');
17187: end;
17188:
17189: procedure SIRegister_RedirFunc(CL: TPPascalCompiler);
17190: begin
17191:   CL.AddTypeS('TPreviousFsRedirectionState', record DidDisable : Boolean; OldValue : __Pointer; end');
17192:   CL.AddDelphiFunction('Function AreFsRedirectionFunctionsAvailable : Boolean');
17193:   Function DisableFsRedirectionIf(const Disable:Boolean;var PreviousState:TPreviousFsRedirectionState):Bool;
17194:   Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState)');
17195:   Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const FIlename : String) : BOOL');
17196:   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;

```

```

17197: Function CopyFileRedir( const DisableFsRedir: Boolean; const ExistingFilename, NewFilename : String; const
17198: FailIfExists : BOOL' );
17199: Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String ) : BOOL' );
17200: Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String ) : Boolean' );
17201: Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String ) : Boolean' );
17202: Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData : TWin32FindData ) : THandle' );
17203: Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String ) : DWORD' );
17204: Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String ) : String' );
17205: Function GetVersionNumbersRedir( const DisableFsRedir: Boolean; const Filename: String; var VersionNumbers : TFileVersionNumbers ) : Boolean' );
17206: Function IsDirectoryAndNotReparsePointRedir( const DisableFsRedir: Boolean; const Name: String ): Bool;
17207: Function MoveFileRedir( const DisableFsRedir: Boolean; const ExistingFilename, NewFilename: String ): Boolean;
17208: Function MoveFileExRedir( const DisableFsRedir: Boolean; const ExistingFile, NewFile: String; const
17209: Flags: DWORD );
17210: Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String ) : Boolean' );
17211: Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String ) : BOOL' );
17212: Function SetFileAttributesRedir( const DisableFsRedir: Boolean; const Filename: String; const Attrib:DWORD ): Boolean;
17213: SIRegister_TFileRedir(CL);
17214: SIRegister_TTextfileReaderRedir(CL);
17215: SIRegister_TTextfileWriterRedir(CL);
17216: end;
17217: procedure SIRegister_Int64Em(CL: TPPascalCompiler);
17218: begin
17219: //CL.AddTypeS('LongWord', 'Cardinal');
17220: CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17221: Function Compare64( const N1, N2 : Integer64 ) : Integer' );
17222: Procedure Dec64( var X : Integer64; N : LongWord );
17223: Procedure Dec6464( var X : Integer64; const N : Integer64 );
17224: Function Div64( var X : Integer64; const Divisor : LongWord ) : LongWord' );
17225: Function Inc64( var X : Integer64; N : LongWord ) : Boolean' );
17226: Function Inc6464( var X : Integer64; const N : Integer64 ) : Boolean' );
17227: Function Integer64ToStr( X : Integer64 ) : String' );
17228: Function Mod64( const X : Integer64; const Divisor : LongWord ) : LongWord' );
17229: Function Mul64( var X : Integer64; N : LongWord ) : Boolean' );
17230: Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64 );
17231: Procedure Shr64( var X : Integer64; Count : LongWord );
17232: Function StrToInteger64( const S : String; var X : Integer64 ) : Boolean' );
17233: end;
17234: 
17235: procedure SIRegister_InstFunc(CL: TPPascalCompiler);
17236: begin
17237: //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17238: SIRegister_TSsimpleStringList(CL);
17239: CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle )');
17240: CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17241: CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte');
17242: CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte');
17243: // TMD5Digest = array[0..15] of Byte;
17244: // TSHA1Digest = array[0..19] of Byte;
17245: Function CheckForMutexes( Mutexes : String ) : Boolean' );
17246: Function CreateTempDir : String' );
17247: Function DecrementSharedCount( const RegView : TRegView; const Filename : String ) : Boolean' );
17248: Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
FirstRetryDelayMS, SubsequentRetryDelayMS : Integer );
17249: //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
TDeleteFileProc; const Param : Pointer );
17250: //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer ) :
TDetermineDefaultLanguageResult );
17251: //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17252: Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String ) : Boolean' );
17253: Function GenerateUniqueName( const DisableFsRedir: Boolean; Path: String; const Extension: String ): String;
17254: Function GetComputerNameString : String' );
17255: Function GetFileDateTime( const DisableFsRedir: Boolean; const Filename: String; var DateTime: TFileTime ): Boolean;
17256: Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String ) : TMD5Digest' );
17257: Function GetMD5OfFileA( const S : AnsiString ) : TMD5Digest' );
17258: // Function GetMD5OfFileU( const S : UnicodeString ) : TMD5Digest' );
17259: Function GetSHA1OfFile( const DisableFsRedir: Boolean; const Filename: String ): TSHA1Digest' );
17260: Function GetSHA1OfFileA( const S : AnsiString ) : TSHA1Digest' );
17261: // Function GetSHA1OfFileU( const S : UnicodeString ) : TSHA1Digest' );
17262: Function GetRegRootKeyName( const RootKey : HKEY ) : String' );
17263: Function GetSpaceOnDisk( const DisableFsRedir: Boolean; const DriveRoot: String; var FreeBytes,
TotBytes: Int64 ) : Boolean;
17264: Function GetSpaceOnNearestMountPoint( const DisableFsRedir: Boolean; const StartDir: String; var FreeBytes,
TotalBytes: Integer64 ) : Boolean;
17265: Function GetUserNamesString : String' );
17266: Procedure IncrementSharedCount( const RegView: TRegView; const Filename: String; const AlreadyExisted: Boolean );
17267: //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir : String;
const Wait:TExecWait; const ShowCmd: Integer; const ProcessMessagesProc: TProcedure; var
ResultCode: Integer );
17268: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
TExecWait; const ShowCmd: Integer; const ProcessMessagesProc: TProcedure; var ResultCode: Integer );
17269: Procedure InternalError( const Id : String );
17270: Procedure InternalErrorFmt( const S : String; const Args : array of const );
17271: Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String ) : Boolean' );

```

```

17272: Function IsProtectedSystemFile( const DisableFsRedir:Boolean; const Filename:String ) : Boolean';
17273: Function MakePendingFileRenameOperationsChecksum : TMD5Digest';
17274: Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean ) : Boolean';
17275: Procedure RaiseFunctionFailedError( const FunctionName : String' );
17276: Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT' );
17277: Procedure RefreshEnvironment' );
17278: Function ReplaceSystemDirWithSysWow64( const Path : String ) : String';
17279: Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean ) : String';
17280: Procedure UnregisterFont( const FontName, FontFilename : String' );
17281: Function RestartComputer : Boolean';
17282: Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String' );
17283: Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String' );
17284: Procedure Win32ErrorMsg( const FunctionName : String' );
17285: Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD' );
17286: Function inForceDirectories( const DisableFsRedir:Boolean; Dir : String ) : Boolean';
17287: //from Func2
17288: //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
17289: //Iconfilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean;
17290: //+'const AppUserID: String; const ExcludeFromShowInNewInstall, PreventPinning: Bool): String');
17291: Procedure RegisterTypeLibrary( const Filename : String' );
17292: //Procedure UnregisterTypeLibrary( const Filename : String' );
17293: function getVersionInfoEx3: TOSVersionInfoEx';
17294: Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean';
17295: procedure InitOle();
17296: Function ExpandConst( const S : String ) : String';
17297: Function ExpandConstEx( const S : String; const CustomConsts : array of String ) : String';
17298: Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17299: Function ExpandConstIfPrefixed( const S : String ) : String';
17300: Procedure LogWindowsVersion';
17301: Function EvalCheck( const Expression : String ) : Boolean';
17302: end;
17303:
17304: procedure SIRegister_unitResourceDetails(CL: TPSPPascalCompiler);
17305: begin
17306: CL.AddClassN(CL.FindClass('TObject'), 'TResourceDetails');
17307: //CL.AddTypes('TResourceDetailsClass', 'class of TResourceDetails');
17308: SIRegister_TResourceModule(CL);
17309: SIRegister_TResourceDetails(CL);
17310: SIRegister_TAnsiResourceDetails(CL);
17311: SIRegister_TUnicodeResourceDetails(CL);
17312: Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass );
17313: Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass );
17314: Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer ) : string';
17315: Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer' );
17316: Function ResourceNameToInt( const s : string ) : Integer';
17317: Function CompareDetails( p1, p2 : TObject(Pointer) ) : Integer';
17318: end;
17319:
17320:
17321: procedure SIRegister_TSsimpleComPort(CL: TPSPPascalCompiler);
17322: begin
17323: //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17324: with CL.AddClassN(CL.FindClass('TObject'), 'TSimpleComPort') do begin
17325: RegisterMethod('Constructor Create');
17326: RegisterMethod('Procedure Free');
17327: RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17328: RegisterMethod('Procedure WriteString( const S : String)');
17329: RegisterMethod('Procedure ReadString( var S : String)');
17330: end;
17331: Ex := SimpleComPort:= TSsimpleComPort.Create;
17332: SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17333: SimpleComPort.WriteString(AsciiChar);
17334: end;
17335:
17336:
17337: procedure SIRegister_Console(CL: TPSPPascalCompiler);
17338: begin
17339: CL.AddConstantN('White','LongInt').SetInt( 15 );
17340: // CL.AddConstantN('Blink','LongInt').SetInt( 128 );
17341: ('conBW40','LongInt').SetInt( 0 );
17342: ('conCO40','LongInt').SetInt( 1 );
17343: ('conBW80','LongInt').SetInt( 2 );
17344: ('conCO80','LongInt').SetInt( 3 );
17345: ('conMono','LongInt').SetInt( 7 );
17346: CL.AddConstantN('conFont8x8','LongInt').SetInt( 256 );
17347: //CL.AddConstantN('C40','','').SetString( C040 );
17348: //CL.AddConstantN('C80','','').SetString( C080 );
17349: Function conReadKey : Char';
17350: Function conKeyPressed : Boolean';
17351: Procedure conGotoXY( X, Y : Smallint );
17352: Function conWhereX : Integer';
17353: Function conWhereY : Integer';
17354: Procedure conTextColor( Color : Byte );
17355: Function conTextColor1 : Byte';
17356: Procedure conTextBackground( Color : Byte );
17357: Function conTextBackground1 : Byte';
17358: Procedure conTextMode( Mode : Word );
17359: Procedure conLowVideo );

```

```

17360: Procedure conHighVideo');
17361: Procedure conNormVideo');
17362: Procedure conClrScr');
17363: Procedure conClrEol');
17364: Procedure conInsLine');
17365: Procedure conDelLine');
17366: Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17367: Function conScreenWidth : Smallint');
17368: Function conScreenHeight : Smallint');
17369: Function conBufferWidth : Smallint');
17370: Function conBufferHeight : Smallint');
17371: procedure InitScreenMode;');
17372: end;
17373:
17374: (*-----*)
17375: procedure SIRegister_testutils(CL: TPSPPascalCompiler);
17376: begin
17377:   SIRegister_TNoRefCountObject(CL);
17378:   Procedure FreeObjects( List : TFPLList );
17379:   Procedure GetMethodList( AObject : TObject; AList : TStrings );
17380:   Procedure GetMethodList1( AClass : TClass; AList : TStrings );
17381: end;
17382:
17383: procedure SIRegister_ToolsUnit(CL: TPSPPascalCompiler);
17384: begin
17385:   'MaxDataSet','LongInt').SetInt( 35 );
17386:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17387:   SIRegister_TDBConnector(CL);
17388:   SIRegister_TDBBasicsTestSetup(CL);
17389:   SIRegister_TTestDataLink(CL);
17390:   'testValuesCount','LongInt').SetInt( 25 );
17391:   Procedure InitialiseDBConnector';
17392:   Procedure FreeDBConnector';
17393:   Function DateTimeToTimeString( d : tdatetime ) : string';
17394:   Function TStringToDateTIme( d : String ) : TDateTIme';
17395: end;
17396:
17397: procedure SIRegister_fpcunit(CL: TPSPPascalCompiler);
17398: begin
17399:   SIRegister_EAssertionFailedError(CL);
17400:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17401:   CL.AddTypeS('TRunMethod', 'Procedure');
17402:   CL.AddClassN(CL.FindClass('TOBJECT'),'TTestResult');
17403:   SIRegister_TTest(CL);
17404:   SIRegister_TAssert(CL);
17405:   SIRegister_TTestFailure(CL);
17406:   SIRegister_ITestlistener(CL);
17407:   SIRegister_TTestCase(CL);
17408:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17409:   SIRegister_TTestSuite(CL);
17410:   SIRegister_TTestResult(CL);
17411:   Function ComparisonMsg( const aExpected : string; const aActual : string ) : string';
17412: end;
17413:
17414: procedure SIRegister_cTCPBuffer(CL: TPSPPascalCompiler);
17415: begin
17416:   TOBJECT', 'ETCPBuffer');
17417:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Integer';
17418:     +r; Head : Integer; Used : Integer; end');
17419:   'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500 );
17420:   'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000 );
17421:   'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBT * 4 );
17422:   'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4 );
17423:   Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int;const TCPBufSize:Int);
17424:   Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer );
17425:   Procedure TCPBufferPack( var TCPBuf : TTCPBuffer );
17426:   Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer );
17427:   Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer );
17428:   Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer );
17429:   Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer ) : Pointer';
17430:   Procedure TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf : string; const Size : Integer );
17431:   Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer ) : Integer';
17432:   Function TCPBufferPeek( var TCPBuf : TTCPBuffer; var Buf : string; const Size:Integer ) : Integer';
17433:   Function TCPBufferRemove( var TCPBuf : TTCPBuffer; var Buf : string; const Size:Integer ) : Integer';
17434:   Procedure TCPBufferClear( var TCPBuf : TTCPBuffer );
17435:   Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer ) : Integer';
17436:   Function TCPBufferUsed( const TCPBuf : TTCPBuffer ) : Integer';
17437:   Function TCPBufferEmpty( const TCPBuf : TTCPBuffer ) : Boolean';
17438:   Function TCPBufferAvailable( const TCPBuf : TTCPBuffer ) : Integer';
17439:   Function TCPBufferPtr( const TCPBuf : TTCPBuffer ) : Pointer';
17440:   Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer );
17441: end;
17442:
17443: procedure SIRegister_Glut(CL: TPSPPascalCompiler);
17444: begin
17445:   //CL.AddTypeS('PInteger', '^Integer // will not work');
17446:   //CL.AddTypeS('PPChar', '^PChar // will not work');
17447:   CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3 );
17448:   CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12 );

```

```

17449: CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0 );
17450: CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5 );
17451: CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6 );
17452: Procedure LoadGlut( const dll : String );
17453: Procedure FreeGlut();
17454: end;
17455:
17456: procedure SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
17457: begin
17458:   CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17459:   Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean) : THandle';
17460: end;
17461:
17462: procedure SIRegister_SwitchLed(CL: TPSPascalCompiler);
17463: begin
17464:   TLedColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black )';
17465:   TTledState', '( LedOn, LedOff, LedDisabled )';
17466:   SIRegister_TSwitchLed(CL);
17467: //CL.AddDelphiFunction('Procedure Register');
17468: end;
17469:
17470: procedure SIRegister_FileClass(CL: TPSPascalCompiler);
17471: begin
17472:   CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17473:     + 'xisting, fdOpenAlways, fdTruncateExisting )');
17474:   CL.AddTypeS('TFileAccess', '( faRead, faWrite, faReadWrite )');
17475:   CL.AddTypeS('TFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17476:   SIRegister_TCustomFile(CL);
17477:   SIRegister_TFile(CL);
17478:   SIRegister_TMemoryFile(CL);
17479:   SIRegister_TTextFileReader(CL);
17480:   SIRegister_TTextFileWriter(CL);
17481:   SIRegister_TFileMapping(CL);
17482:   SIRegister_EFileError(CL);
17483: end;
17484:
17485: procedure SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
17486: begin
17487:   CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17488:     + ', ffaDirectory, ffaArchive, ffaAnyFile )');
17489:   CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17490:   SIRegister_TFileSearch(CL);
17491: end;
17492:
17493: procedure SIRegister_uColorFunctions(CL: TPSPascalCompiler);
17494: begin
17495:   TRGBTYpe', 'record RedHex : string; GreenHex : string; BlueHex :'
17496:     + ' string; Red : integer; Green : integer; Blue : integer; end');
17497:   Function FadeColor( aColor : Longint; aFade : integer) : Tcolor';
17498:   Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17499: end;
17500:
17501: procedure SIRegister_uSettings(CL: TPSPascalCompiler);
17502: begin
17503:   Procedure SaveOscSettings());
17504:   Procedure GetOscSettings());
17505: end;
17506:
17507: procedure SIRegister_cyDebug(CL: TPSPascalCompiler);
17508: begin
17509:   TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer )';
17510:   RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17511:     FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17512:     64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17513:     Cardinal; EnterCount : Integer; ExitCount : Integer; end');
17514:   SIRegister_TcyDebug(CL);
17515: end;
17516:
17517: (*-----*)
17518: procedure SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
17519: begin
17520:   TCopyFileResult', '( cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )';
17521:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )';
17522:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )';
17523:   SIRegister_TDestinationOptions(CL);
17524:   TProcCustomCopyFileEvent', 'Procedure ( Sender : TObject; var CopyFileResult : TCopyFileResult )';
17525:   TProcOnCopyFileProgressEvent', 'Procedure(Sender:TObject;FileBytes,
TransferredBytes:int64;PercentDone:Int64)';
17526:   TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String )';
17527:   SIRegister_TcyCopyFiles(CL);
17528:   Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
ResetAttr:boolean): TCopyFileResult';
17529:   Function cyCopyFileEx( Fromfile,ToFile: String;FileExists: TCopyFileExists;FileNotExists:TCopyFileNotExists
TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult';
17530:   Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;
+ FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer';
17531: end;
17532:
17533:

```

```

17534: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17535: begin
17536:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17537:   SIRegister_TcyFileAttributes(CL);
17538:   SIRegister_TSearchRecInstance(CL);
17539:   TOption', '( soOnlyDirs, soignoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17540:   TOptions', 'set of TOption');
17541:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17542:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
Accept:boolean;
17543:   TProcOnValidateDirectoryEvent','Procedure(Sender:TObject;Directory:String;var Accept:boolean)');
17544:   SIRegister_TcyCustomSearchFiles(CL);
17545:   SIRegister_TcySearchFiles(CL);
17546:   Function FileNameRespondToMask( aFileName : String; aMask : String ) : Boolean');
17547:   Function IscyFolder( aSRec : TSearchrec ) : Boolean');
17548: end;
17549:
17550: procedure SIRegister_jcontrolutils(CL: TPSPascalCompiler);
17551: begin
17552:   Function jCountChar( const s : string; ch : char ) : integer');
17553:   Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings );
17554:   Function jNormalizeDate(const Value: string; theValue: TDateTime;const theFormat:string): string');
17555:   Function jNormalizeTime(const Value: string; theValue: TTime;const theFormat : string) : string');
17556:   Function jNormalizeDateTime(const Value:stringtheValue:TDateTime;const theFormat:string):string');
17557:   Function jNormalizeDateSeparator( const s : string ) : string');
17558:   Function jiIsValidDateString( const Value : string ) : boolean');
17559:   Function jiIsValidTimeString( const Value : string ) : boolean');
17560:   Function jiIsValidDateTimeString( const Value : string ) : boolean');
17561: end;
17562:
17563: procedure SIRegister_kcMapView(CL: TPSPascalCompiler);
17564: begin
17565:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMapView');
17566:   CL.AddTypeS('TMapSource', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo' +
'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik' +
'+, msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual' +
'+EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa' +
'+hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17567:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17568:   TRealArea', 'record top : Extended; left : Extended; bottom : Extended; right : Extended; end');
17569:   TIntPoint', 'record X : Int64; Y : Int64; end');
17570:   TkRealPoint', 'record X : Extended; Y : Extended; end');
17571:   TOnBeforeDownloadEvent', 'Procedure ( Url : string; str : TStream; var CanHandle : Boolean)');
17572:   TOnAfterDownloadEvent', 'Procedure ( Url : string; str : TStream)');
17573:   SIRegister_TCustomDownloadEngine(CL);
17574:   SIRegister_TCustomGeolocationEngine(CL);
17575:   SIRegister_TMapView(CL);
17576: end;
17577:
17578: procedure SIRegister_cparserutils(CL: TPSPascalCompiler);
17579: begin
17580:   (*CL.AddDelphiFunction('Function isFunc( name : TNamePart ) : Boolean');*)
17581:   CL.AddDelphiFunction('Function isUnnamedFunc( name : TNamepart ) : Boolean');
17582:   Function isPtrToFunc( name : TNamePart ) : Boolean');
17583:   Function isFuncRetFuncPtr( name : TNamePart ) : Boolean');
17584:   Function isPtrToFuncRetFuncPtr( name : TNamePart ) : Boolean');
17585:   Function GetFuncParam( name : TNamePart ) : TNamePart');
17586:   Function isArray( name : TNamePart ) : Boolean');
17587:   Function GetArrayPart( name : TNamePart ) : TNamePart');
17588:   Function GetIdFromPart( name : TNamePart ) : AnsiString');
17589:   Function GetIdPart( name : TNamePart ) : TNamePart');
17590:   Function isNamePartPtrToFunc( part : TNamePart ) : Boolean');
17591:   Function isAnyBlock( part : TNamePart ) : Boolean');*
17592:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17593:   SIRegister_TLineBreaker(CL);
17594:   CL.AddTypeS('TNameKind', 'Integer');
17595:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNamePart');
17596:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17597:   Function SphericalMod( X : Extended ) : Extended');
17598:   Function cSign( Value : Extended ) : Extended');
17599:   Function LimitFloat( const eValue, eMin, eMax : Extended ) : Extended');
17600:   Function AngleToRadians( iAngle : Extended ) : Extended');
17601:   Function RadiansToAngle( eRad : Extended ) : Extended');
17602:   Function Cross180( iLong : Double ) : Boolean');
17603:   Function Mod180( Value : integer ) : Integer');
17604:   Function Mod180Float( Value : Extended ) : Extended');
17605:   Function MulDivFloat( a, b, d : Extended ) : Extended');
17606:   Function LongDiff( ilong1, ilong2 : Double ) : Double');
17607:   Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap );
17608:   Function Bmp_CreateFromPersistent( Source : TPersistent ) : TbitMap');
17609:   Function FixFilePath( const Inpath, CheckPath : string ) : string');
17610:   Function UnFixFilePath( const Inpath, CheckPath : string ) : string');
17611:   Procedure FillStringList( sl : TStringList; const aText : string );
17612: end;
17613:
17614: procedure SIRegister_LedNumber(CL: TPSPascalCompiler);
17615: begin
17616:   TLedSegmentSize', 'Integer');
17617:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised )');

```

```

17622:   SIRegister_TCustomLEDNumber(CL);
17623:   SIRegister_TLEDNumber(CL);
17624: end;
17625;
17626: procedure SIRegister_StStrL(CL: TPSPPascalCompiler);
17627: begin
17628:   CL.AddTypeS('LStrRec', 'record AllocSize : Longint; RefCount : Longint; Length : Longint; end');
17629:   CL.AddTypes('AnsiChar', 'Char');
17630:   CL.AddTypeS('BTable', 'array[0..255] of Byte'); //!!!
17631:   CL.AddDelphiFunction('Function HexBL( B : Byte) : AnsiString');
17632:   Function HexWL( W : Word) : AnsiString';
17633:   Function HexLL( L : LongInt) : AnsiString';
17634:   Function HexPTRL( P : __Pointer) : AnsiString';
17635:   Function BinaryBL( B : Byte) : AnsiString';
17636:   Function BinaryWL( W : Word) : AnsiString';
17637:   Function BinaryLL( L : LongInt) : AnsiString';
17638:   Function OctalBL( B : Byte) : AnsiString';
17639:   Function OctalWL( W : Word) : AnsiString';
17640:   Function OctalLL( L : LongInt) : AnsiString';
17641:   Function Str2Int16L( const S : AnsiString; var I : SmallInt) : Boolean';
17642:   Function Str2WordL( const S : AnsiString; var I : Word) : Boolean';
17643:   Function Str2LongL( const S : AnsiString; var I : LongInt) : Boolean';
17644:   Function Str2RealL( const S : AnsiString; var R : Double) : Boolean';
17645:   Function Str2RealL( const S : AnsiString; var R : Real) : Boolean';
17646:   Function Str2ExtL( const S : AnsiString; var R : Extended) : Boolean';
17647:   Function Long2StrL( L : LongInt) : AnsiString';
17648:   Function Real2StrL( R : Double; Width : Byte; Places : ShortInt) : AnsiString';
17649:   Function Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : AnsiString';
17650:   Function ValPrepL( const S : AnsiString) : AnsiString';
17651:   Function CharStrL( C : Char; Len : Cardinal) : AnsiString';
17652:   Function PadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString';
17653:   Function PadLL( const S : AnsiString; Len : Cardinal) : AnsiString';
17654:   Function LeftPadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString';
17655:   Function LeftPadL( const S : AnsiString; Len : Cardinal) : AnsiString';
17656:   Function TrimLeadL( const S : AnsiString) : AnsiString';
17657:   Function TrimTrailL( const S : AnsiString) : AnsiString';
17658:   Function TrimL( const S : AnsiString) : AnsiString';
17659:   Function TrimSpacesL( const S : AnsiString) : AnsiString';
17660:   Function CenterChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString';
17661:   Function CenterL( const S : AnsiString; Len : Cardinal) : AnsiString';
17662:   Function EntabL( const S : AnsiString; TabSize : Byte) : AnsiString';
17663:   Function DetabL( const S : AnsiString; TabSize : Byte) : AnsiString';
17664:   Function ScrambleL( const S, Key : AnsiString) : AnsiString';
17665:   Function SubstituteL( const S, FromStr,ToStr : AnsiString) : AnsiString';
17666:   Function FilterL( const S, Filters : AnsiString) : AnsiString';
17667:   Function CharExistsL( const S : AnsiString; C : AnsiChar) : Boolean';
17668:   Function CharCountL( const S : AnsiString; C : AnsiChar) : Cardinal';
17669:   Function WordCountL( const S, WordDelims : AnsiString) : Cardinal';
17670:   Function WordPositionL( N : Cardinal; const S, WordDelims : AnsiString; var Pos : Cardinal) : Boolean';
17671:   Function ExtractWordL( N : Cardinal; const S, WordDelims : AnsiString) : AnsiString';
17672:   Function AsciiCountL( const S, WordDelims : AnsiString; Quote : AnsiChar) : Cardinal';
17673:   Function AsciiPositionL(N: Cardinal;const S,WordDelims:AnsiString;Quote:AnsiChar;var Pos:Cardinal):Bool;
17674:   Function ExtractAsciiL( N : Cardinal; const S, WordDelims : AnsiString; Quote : AnsiChar) : AnsiString';
17675:   Procedure WordWrapL(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17676:   Procedure WordWrap(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17677:   Function CompStringL( const S1, S2 : AnsiString) : Integer';
17678:   Function CompUCStringL( const S1, S2 : AnsiString) : Integer';
17679:   Function SoundexL( const S : AnsiString) : AnsiString';
17680:   Function MakeLetterSetL( const S : AnsiString) : Longint';
17681:   Procedure BMMakeTableL( const MatchString : AnsiString; var BT : BTable)');
17682:   Function BMSearchL(var Buffer,BufLength:Cardinal;var BT:BTable;const MatchString:AnsiString;var
Pos:Card):Bool;
17683:   Function BMSearchUCL( var Buffer, BufLength : Cardinal; var BT : BTable; const MatchString : AnsiString;
var Pos : Cardinal) : Boolean';
17684:   Function DefaultExtensionL( const Name, Ext : AnsiString) : AnsiString';
17685:   Function ForceExtensionL( const Name, Ext : AnsiString) : AnsiString';
17686:   Function JustFilenameL( const PathName : AnsiString) : AnsiString';
17687:   Function JustNameL( const PathName : AnsiString) : AnsiString';
17688:   Function JustExtensionL( const Name : AnsiString) : AnsiString';
17689:   Function JustPathnameL( const PathName : AnsiString) : AnsiString';
17690:   Function AddBackSlashL( const DirName : AnsiString) : AnsiString';
17691:   Function CleanPathNameL( const PathName : AnsiString) : AnsiString';
17692:   Function HasExtensionL( const Name : AnsiString; var DotPos : Cardinal) : Boolean';
17693:   Function CommaizeL( L : Longint) : AnsiString';
17694:   Function CommaizeChL( L : Longint; Ch : AnsiChar) : AnsiString';
17695:   Function FloatFormL(const Mask:AnsiString;R:TstFloat;const LtCurr,RtCurr:AnsiString;Sep,
DecPt:Char):AnsiString;
17696:   Function LongIntFormL(const Mask:AnsiString;L:LongInt;const LtCurr,
RtCurr:AnsiString;Sep:Char):AnsiString';
17697:   Function StrChPosL( const P : AnsiString; C : AnsiChar; var Pos : Cardinal) : Boolean';
17698:   Function StrStPosL( const P, S : AnsiString; var Pos : Cardinal) : Boolean';
17699:   Function StrStCopyL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString';
17700:   Function StrChInsertL( const S : AnsiString; C : AnsiChar; Pos : Cardinal) : AnsiString';
17701:   Function StrStInsertL( const S1, S2 : AnsiString; Pos : Cardinal) : AnsiString';
17702:   Function StrChDeleteL( const S : AnsiString; Pos : Cardinal) : AnsiString';
17703:   Function StrStDeleteL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString';
17704:   Function ContainsOnlyL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean';
17705:   Function ContainsOtherThanL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean';
17706:   Function CopyLeftL( const S : AnsiString; Len : Cardinal) : AnsiString';

```

```

17707: Function CopyMidL( const S : AnsiString; First, Len : Cardinal ) : AnsiString');
17708: Function CopyRightL( const S : AnsiString; First : Cardinal ) : AnsiString');
17709: Function CopyRightAbsL( const S : AnsiString; NumChars : Cardinal ) : AnsiString');
17710: Function CopyFromNthWordL( const S,WordDelims:AString;const AWord:AString;N:Cardinal;var
SubString:AString):Bool;
17711: Function CopyFromToWordL(const S,WordDelims,Word1,Word2:AnsiString;N1,N2:Cardinal;var
SubString:AnsiString):Bool;
17712: Function CopyWithinL( const S, Delimiter : AnsiString; Strip : Boolean ) : AnsiString');
17713: Function DeleteFromNthWordL( const S, WordDelims : AnsiString; const AWord : AnsiString; N : Cardinal;
var SubString : AnsiString ) : Boolean');
17714: Function DeleteFromToWordL( const S, WordDelims, Word1, Word2 : AnsiString; N1, N2 : Cardinal; var
SubString : AnsiString ) : Boolean');
17715: Function DeleteWithinL( const S, Delimiter : AnsiString ) : AnsiString');
17716: Function ExtractTokensL( const S,
Delims:AnsiString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Cardinal;
17717: Function IsChAlphaL( C : AnsiChar ) : Boolean');
17718: Function IsChNumericL( C : AnsiChar; const Numbers : AnsiString ) : Boolean');
17719: Function IsChAlphaNumericL( C : AnsiChar; const Numbers : AnsiString ) : Boolean');
17720: Function IsStrAlphaL( const S : AnsiString ) : Boolean');
17721: Function IsStrNumericL( const S, Numbers : AnsiString ) : Boolean');
17722: Function IsStrAlphaNumericL( const S, Numbers : AnsiString ) : Boolean');
17723: Function KeepCharsL( const S, Chars : AnsiString ) : AnsiString');
17724: Function LastWordL( const S, WordDelims, AWord : AnsiString; var Position : Cardinal ) : Boolean');
17725: Function LastWordAbsL( const S, WordDelims : AnsiString; var Position : Cardinal ) : Boolean');
17726: Function LastStringL( const S, AString : AnsiString; var Position : Cardinal ) : Boolean');
17727: Function LeftTrimCharsL( const S, Chars : AnsiString ) : AnsiString');
17728: Function ReplaceWordL(const S, WordDelims,OldWord,NewWord:AnsiString;N:Cardinal;var
Replacements:Cardinal):AnsiString;
17729: Function ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:AnsiString;var
Replacements:Cardinal):AnsiString');
17730: Function ReplaceStringL(const S,OldString,NewString:AnsiString;N:Cardinal;var
Replacements:Cardinal):AnsiString;
17731: Function ReplaceStringAllL(const S,OldString,NewString:AnsiString; var Replacements:Cardinal):AnsiString;
17732: Function RepeatStringL(const RepeatString:AnsiString;var Repetitions:Cardinal;MaxLen:Cardinal):AnsiString;
17733: Function RightTrimCharsL( const S, Chars : AnsiString ) : AnsiString');
17734: Function StrWithinL( const S, SearchStr : string; Start : Cardinal; var Position : Cardinal ) : boolean');
17735: Function TrimCharsL( const S, Chars : AnsiString ) : AnsiString');
17736: Function WordPosL(const S,WordDelims,AWord: AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17737: Function WordPos(const S,WordDelims,AWord:AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17738: end;
17739:
17740: procedure SIRegister_pwnative_out(CL: TPSPPascalCompiler);
17741: begin
17742: CL.AddConstantN('STDIN','LongInt').SetInt( 0 );
17743: ('STDOUT','LongInt').SetInt( 1 );
17744: ('STDERR','LongInt').SetInt( 2 );
17745: Procedure NativeWrite( s : astr);';
17746: Procedure NativeWriteln( PString : PChar );';
17747: Procedure NativeWrite2( Buffer : PChar; NumChars : Cardinal );';
17748: Procedure NativeWriteLn( s : astr);';
17749: Procedure NativeWriteLn1();';
17750: end;
17751:
17752: procedure SIRegister_synwrap1(CL: TPSPPascalCompiler);
17753: begin
17754: CL.AddTypeS(TSynwInfo', 'record Err : byte; UrlHtml : ansistring; ErrResponse
17755: : integer; UltimateURL : ansistring; Headers : ansistring; end');
17756: CL.AddTypeS('TUrlInfo', 'record Err : byte; UltimateURL : string; end');
17757: Function GetHttpFile( const Url, UserAgent, Outfile : string; verbose : boolean): TSynwInfo;');
17758: Function GetHttpFile1( const Url, UserAgent, Outfile : string ) : TSynwInfo;');
17759: Function GetHttpFile2( const Url, Outfile : string ) : TSynwInfo;');
17760: Function GetHttpFile3( const Url, outfile : string; verbose : boolean) : TSynwInfo;');
17761: Function GetHtm( const Url : string ) : string;');
17762: Function GetHtml( const Url, UserAgent : string ) : string;');
17763: Function GetUrl( const Url : string; verbose : boolean) : TSynwInfo;');
17764: Function GetUrl1( const Url, useragent : string ) : TSynwInfo;');
17765: Function GetUrl2( const Url : string ) : TSynwInfo;');
17766: Function GetUrl3( const Url : string; const http : THTTPPSend; verbose : boolean): TUrlInfo;');
17767: Function GetUrl4( const Url : string; const http : THTTPPSend ) : TUrlInfo;');
17768: Procedure StrToStream( s : String; strm : TMemoryStream );
17769: Function StrLoadStream( strm : TStream ) : String');
17770:
17771: end;
17772:
17773: procedure SIRegister_HTMLUtil(CL: TPSPPascalCompiler);
17774: begin
17775: Function GetVal( const tag, attribname_ci : string ) : string');
17776: Function GetTagName( const Tag : string ) : string');
17777: Function GetUpTagName( const tag : string ) : string');
17778: Function GetNameValPair( const tag, attribname_ci : string ) : string');
17779: Function GetValFromNameVal( const namevalpair : string ) : string');
17780: Function GetNameValPair_cs( const tag, attribname : string ) : string');
17781: Function GetVal_JAMES( const tag, attribname_ci : string ) : string');
17782: Function GetNameValPair_JAMES( const tag, attribname_ci : string ) : string');
17783: Function CopyBuffer( StartIndex : PChar; Len : integer ) : string');
17784: Function Ucase( s : string ) : string');
17785: end;
17786:
17787: procedure SIRegister_pwmain(CL: TPSPPascalCompiler);

```

```

17788: begin
17789:   CL.AddConstantN('FUTURE_COOKIE','String').SetString('Mon, 01 Dec 2099 12:00:00 GMT');
17790:   EXPIRED_COOKIE,'String').SetString('Mon, 01 Jan 2001 12:00:00 GMT');
17791:   'SECURE_OFF','LongInt').SetInt( 0);
17792:   'SECURE_ON','LongInt').SetInt( 2);
17793:   'SECURE_FILTER','LongInt').SetInt( 3);
17794:   THandle or DWord!
17795: //  astr = ansistring;
17796: CL.AddTypeS('pastr', 'ansistring');
17797: CL.AddTypeS('TFilterFunc', 'function(const s: pastr): pastr;');
17798: uses pwinit at begin
17799: //type TFilterFunc = function(const s: astr): astr;
17800: Demo: ..\maxbox3\examples2\519_pwtills.txt
17801:
17802: //CL.AddConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false);
17803: //CL.AddConstantN('CASE_IGNORE','Boolean')BoolToStr( false);
17804: Procedure pwInit';
17805: Procedure OffReadln';
17806: Function Lcase( const s : pastr) : pastr';
17807: Function Ucase( const s : pastr) : pastr';
17808: Function CountPostVars : longword';
17809: Function GetPostVar( const name : pastr) : pastr;');
17810: Function GetPostVar1( const name : pastr; filter : TFilterFunc) : pastr;');
17811: Function GetPostVar_S( const name : pastr; Security : integer) : pastr');
17812: Function GetPostVar_SF( const name : pastr; Security : integer) : pastr');
17813: Function GetPostVarAsFloat( const name : pastr) : double');
17814: Function GetPostVarAsInt( const name : pastr) : longint');
17815: Function GetPostVar_SafeHTML( const name : pastr) : pastr');
17816: Function FetchPostVarName( idx : longword) : pastr');
17817: Function FetchPostVarVal( idx : longword) : pastr');
17818: Function FetchPostVarVal1( idx : longword; filter : TFilterFunc) : pastr;');
17819: Function FetchPostVarName_S( idx : longword; Security : integer) : pastr');
17820: Function FetchPostVarVal_S( idx : longword; Security : integer) : pastr');
17821: Function IsPostVar( const name : pastr) : boolean');
17822: Function CountAny : longword');
17823: Function GetAny( const name : pastr) : pastr');
17824: Function GetAny1( const name : pastr; filter : TFilterFunc) : pastr');
17825: Function GetAny_S( const name : pastr; Security : integer) : pastr');
17826: Function GetAnyAsFloat( const name : pastr) : double');
17827: Function GetAnyAsInt( const name : pastr) : longint');
17828: Function IsAny( const name : pastr) : byte');
17829: Function CountCookies : longword');
17830: Function FetchCookieName( idx : longword) : pastr');
17831: Function FetchCookieVal( idx : longword) : pastr');
17832: Function FetchCookieVal1( idx : longword; filter : TFilterFunc) : pastr');
17833: Function GetCookie( const name : pastr) : pastr');
17834: Function GetCookie1( const name : pastr; filter : TFilterFunc) : pastr');
17835: Function GetCookieAsFloat( const name : pastr) : double');
17836: Function GetCookieAsInt( const name : pastr) : longint');
17837: Function IsCookie( const name : pastr) : boolean');
17838: Function SetCookie( const name, value : pastr) : boolean');
17839: Function SetCookieAsFloat( const name : pastr; value : double) : boolean');
17840: Function SetCookieAsInt( const name : pastr; value : longint) : boolean');
17841: Function SetCookieEx( const name, value, path, domain, expiry : pastr) : boolean');
17842: Function SetCookieAsFloatEx( const name:pastr;value : double; const path, domain, expiry:pastr):bool;
17843: Function SetCookieAsIntEx( const name:pastr;value : longint; const path, domain, expiry:pastr):bool;
17844: Function UnsetCookie( const name : pastr) : boolean');
17845: Function UnsetCookieEx( const name, path, domain : pastr) : boolean');
17846: Function FilterHtml( const input : pastr) : pastr');
17847: Function FilterHtml_S( const input : pastr; security : integer) : pastr');
17848: Function TrimBadChars( const input : pastr) : pastr');
17849: Function TrimBadFile( const input : pastr) : pastr');
17850: Function TrimBadDir( const input : pastr) : pastr');
17851: Function TrimBad_S( const input : pastr; security : integer) : pastr');
17852: Function CountHeaders : longword');
17853: Function FetchHeaderName( idx : longword) : pastr');
17854: Function FetchHeaderVal( idx : longword) : pastr');
17855: Function GetHeader( const name : pastr) : pastr');
17856: Function IsHeader( const name : pastr) : boolean');
17857: Function SetHeader( const name, value : pastr) : boolean');
17858: Function UnsetHeader( const name : pastr) : boolean');
17859: Function PutHeader( const header : pastr) : boolean');
17860: Procedure OutL( const s : pastr)');
17861: Procedure OutLn( const s : pastr)');
17862: Procedure OutA( args : array of const)');
17863: Procedure OutF( const s : pastr)');
17864: Procedure OutLnF( const s : pastr)');
17865: Procedure OutFF( const s : pastr)');
17866: Procedure OutF_FI( const s : pastr; HTMLFilter : boolean)');
17867: Procedure OutLnFF( const s : pastr)');
17868: Procedure OutLnF_FI( const s : pastr; HTMLFilter : boolean)');
17869: Function FileOut( const fname : pastr) : word');
17870: Function ResourceOut( const fname : pastr) : word');
17871: Procedure BufferOut( const buff, len : LongWord)');
17872: Function TemplateOut( const fname : pastr; HtmlFilter : boolean) : word;');
17873: Function TemplateOut1( const fname : pastr) : word;');
17874: Function TemplateOut2( const fname : pastr; filter : TFilterFunc) : word;');
17875: Function TemplateOut3( const fname : pastr; HtmlFilter : boolean) : word;');
17876: Function TemplateRaw( const fname : pastr) : word');

```

```

17877: Function Fmt( const s : pastr ) : pastr;');
17878: Function Fmtl( const s : pastr; filter : TFilterFunc ) : pastr;');
17879: Function FmtFilter( const s : pastr ) : pastr');
17880: Function Fmt_SF( const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecurity:int):pastr;
17881: Function Fmt_SF1( const s:pastr; HTMLFilter:boolean; FilterSecurity , TrimSecurity : integer ) : pastr;');
17882: Function CountRtiVars : longword');
17883: Function FetchRtiName( idx : longword ) : pastr');
17884: Function FetchRtiVal( idx : longword ) : pastr');
17885: Function GetRti( const name : pastr ) : pastr');
17886: Function GetRtiAsFloat( const name : pastr ) : double');
17887: Function GetRtiAsInt( const name : pastr ) : longint');
17888: Function IsRti( const name : pastr ) : boolean');
17889: Procedure SetRTI( const name, value : pastr)');
17890: Function FetchUpfileName( idx : longword ) : pastr');
17891: Function GetUpfileName( const name : pastr ) : pastr');
17892: Function GetUpfileSize( const name : pastr ) : longint');
17893: Function GetUpFileType( const name : pastr ) : pastr');
17894: Function CountUpfiles : longword');
17895: Function IsUpfile( const name : pastr ) : boolean');
17896: Function SaveUpfile( const name, fname : pastr ) : boolean');
17897: Function CountVars : longword');
17898: Function FetchVarName( idx : longword ) : pastr');
17899: Function FetchVarVal( idx : longword ) : pastr');
17900: Function FetchVarVal( idx : longword; filter : TFilterFunc ) : pastr');
17901: Function GetVar( const name : pastr ) : pastr');
17902: Function GetVarL( const name : pastr; filter : TFilterFunc ) : pastr');
17903: Function GetVar_S( const name : pastr; security : integer ) : pastr');
17904: Function GetVarAsFloat( const name : pastr ) : double');
17905: Function GetVarAsInt( const name : pastr ) : longint');
17906: Procedure SetVar( const name, value : pastr)');
17907: Procedure SetVarAsFloat( const name : pastr; value : double)');
17908: Procedure SetVarAsInt( const name : pastr; value : longint)');
17909: Function IsVar( const name : pastr ) : byte');
17910: Procedure UnsetVar( const name : pastr)');
17911: Function LineEndToBR( const s : pastr ) : pastr');
17912: Function RandomStr( len : longint ) : pastr');
17913: Function XorCrypt( const s : pastr; key : byte ) : pastr');
17914: Function CountCfgVars : longword');
17915: Function FetchCfgVarName( idx : longword ) : pastr');
17916: Function FetchCfgVarVal( idx : longword ) : pastr');
17917: Function IsCfgVar( const name : pastr ) : boolean');
17918: Function SetCfgVar( const name, value : pastr ) : boolean');
17919: Function GetCfgVar( const name : pastr ) : pastr');
17920: Procedure ThrowErr( const s : pastr );
17921: Procedure ThrowWarn( const s : pastr );
17922: Procedure ErrWithHeader( const s : pastr );
17923: CL.AddTypeS('TWebVar', 'record name : pastr; value : pastr; end');
17924: CL.AddTypeS('TWebVars', 'array of TWebVar');
17925: Function iUpdateWebVar(var webv : TWebVars; const name, value:pastr; upcased:boolean):boolean';
17926: Function iAddWebCfgVar( const name, value : pastr ) : boolean');
17927: Procedure iAddWebVar( var webv : TWebVars; const name, value : pastr );
17928: Procedure iSetRTI( const name, value : pastr );
17929: Function iCustomSessUnitSet : boolean );
17930: Function iCustomCfgUnitSet : boolean );
17931: end;
17932:
17933: procedure SIRegister_W32VersionInfo(CL: TPPSPascalCompiler);
17934: begin
17935:   SIRegister_TProjectVersionInfo(CL);
17936:   CL.AddDelphiFunction('Function MSLanguageToHex( const s : string ) : string');
17937:   Function MSHexToLanguage( const s : string ) : string');
17938:   Function MSCharacterSetToHex( const s : string ) : string');
17939:   Function MSHexToCharacterSet( const s : string ) : string');
17940:   Function MSLanguages : TStringList');
17941:   Function MSHexLanguages : TStringList');
17942:   Function MSCharacterSets : TStringList');
17943:   Function MSHexCharacterSets : TStringList');
17944: end;
17945:
17946: procedure SIRegister_IpUtils(CL: TPPSPascalCompiler);
17947: begin
17948:   TIpHandle', 'Cardinal');
17949:   TIpMD5StateArray', 'array[0..3] of DWORD');
17950:   CL.AddTypeS('TIpMD5CountArray', 'array[0..1] of DWORD');
17951:   TIpMD5ByteBuf', 'array[0..63] of Byte');
17952:   TIpMD5LongBuf', 'array[0..15] of DWORD');
17953:   TIpMD5Digest', 'array[0..15] of Byte');
17954:   TIpLineTerminator', '( ltNone, ltCR, ltLF, ltCRLF, ltOther )');
17955:   TIpMD5Context', 'record State : TIpMD5StateArray; Count : TIpMD5';
17956:     +'CountArray; ByteBuf : TIpMD5ByteBuf; end');
17957:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EIipBaseException');
17958:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EIipAccessException');
17959:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EIipHtmlException');
17960:   SIRegister_TIpBaseAccess(CL);
17961:   SIRegister_TIpBasePersistent(CL);
17962:   //TIPComponentClass', 'class of TIpBaseComponent');
17963:   SIRegister_TIpBaseComponent(CL);
17964:   SIRegister_TIpBaseWinControl(CL);
17965:   Function InClassA( Addr : LongInt ) : Boolean';

```

```

17966: Function InClassB( Addr : LongInt ) : Boolean';
17967: Function InClassC( Addr : LongInt ) : Boolean';
17968: Function InClassD( Addr : LongInt ) : Boolean';
17969: Function InMulticast( Addr : LongInt ) : Boolean';
17970: Function IpCharCount( const Buffer, BufSize : DWORD; C : AnsiChar ) : DWORD';
17971: Function IpCompStruct( const S1, S2, Size : Cardinal ) : Integer';
17972: Function IpMaxInt( A, B : Integer ) : Integer';
17973: Function IpMinInt( A, B : Integer ) : Integer';
17974: Procedure IpSafeFree( var Obj: TObject );
17975: Function IpShortVersion : string';
17976: Function IpInternetSumPrim( var Data, DataSize, CurCrc : DWORD ) : DWORD';
17977: Function IpInternetSumOfStream( Stream : TStream; CurCrc : DWORD ) : DWORD';
17978: Function IpInternetSumOfFile( const FileName : string ) : DWORD';
17979: Function MD5SumOfFile( const FileName : string ) : string';
17980: Function MD5SumOfStream( Stream : TStream ) : string';
17981: Function MD5SumOfStreamDigest( Stream : TStream ) : TIPMD5Digest';
17982: Function MD5SumOfString( const S : string ) : string';
17983: Function MD5SumOfStringDigest( const S : string ) : TIPMD5Digest';
17984: Function SafeYield : LongInt';
17985: Function AllTrimSpaces( Strng : string ) : string';
17986: Function IpCharPos( C : AnsiChar; const S : string ) : Integer';
17987: Function CharPosIdx( C : AnsiChar; const S : string; Idx : Integer ) : Integer';
17988: Function NthCharPos( C : AnsiChar; const S : string; Nth : Integer ) : Integer';
17989: Function RCharPos( C : AnsiChar; const S : string ) : Integer';
17990: Function RCharPosIdx( C : AnsiChar; const S : string; Idx : Integer ) : Integer';
17991: Function RNthCharPos( C : AnsiChar; const S : string; Nth : Integer ) : Integer';
17992: Function IpRPos( const Substr : string; const S : string ) : Integer';
17993: Function IpPosIdx( const SubStr, S : string; Idx : Integer ) : Integer';
17994: ACharSet', 'set of AnsiChar');
17995: TIPAddrRec', 'record Scheme : string; UserName : string; Password string; Authority : string;
17996: Port : string; Path : string; Fragment : string; Query : string; QueryDelim : AnsiChar; end');
17997: Procedure Initialize( var AddrRec : TIPAddrRec );
17998: Procedure Finalize( var AddrRec : TIPAddrRec );
17999: Function ExtractEntityName( const NamePath : string ) : string';
18000: Function ExtractEntityPath( const NamePath : string ) : string';
18001: Function IpParseURL( const URL : string; var Rslt : TIPAddrRec ) : Boolean';
18002: Function BuildURL( const OldURL, NewURL : string ) : string';
18003: Function PutEscapes( const S : string; EscapeSet : ACharSet ) : string';
18004: Function RemoveEscapes( const S : string; EscapeSet : ACharSet ) : string';
18005: Procedure SplitParams( const Params : string; Dest : TStrings );
18006: Function NetToDOSPath( const PathStr : string ) : string';
18007: Function DOSToNetPath( const PathStr : string ) : string';
18008: Procedure SplitHttpResponse( const S : string; var V, MsgID, Msg : string );
18009: Procedure FieldFix( Fields : TStrings );
18010: Function AppendSlash( APath : string ) : string';
18011: Function RemoveSlash( APath : string ) : string';
18012: Function GetParentPath( const Path : string ) : string';
18013: Function GetLocalContent( const TheFileName : string ) : string';
18014: Function IPDirExists( Dir : string ) : Boolean';
18015: Function GetTemporaryFile( const Path : string ) : string';
18016: Function GetTemporaryPath : string');
18017: Function AppendBackSlash( APath : string ) : string';
18018: Function IpRemoveBackSlash( APath : string ) : string';
18019: Function INetDateToStrToDate( const DateStr : string ) : TDateTime';
18020: Function DateToINetDateToStr( DateTime : TDateTime ) : string';
18021: Function IpTimeZoneBias : Integer );
18022: Procedure SplitCookieFields( const Data : string; Fields : TStrings );
18023: end;
18024:
18025: procedure SIRegister_LrtPoTools(CL: TPSPascalCompiler);
18026: begin
18027:   CL.AddTypeS('TPOStyle', '( postStandard, postPropertyName, postFull )');
18028:   Procedure Lrt2Po( const LRTfile : string; POStyle : TPOStyle );
18029:   Procedure CombinePoFiles( SL : TStrings; const FName : string );
18030: end;
18031:
18032: procedure SIRegister_GPS(CL: TPSPascalCompiler);
18033: begin
18034:   CL.AddConstantN('MAX_SATS','LongInt').SetInt( 12 );
18035:   GPSMSG_START', 'String').SetString( '$' );
18036:   GPSMSG_STOP', 'String').SetString( '*' );
18037:   SEC_BETWEEN_SEG', 'LongInt').SetInt( 5 );
18038:   CL.AddTypeS('TSatellite', 'record Identification : Shortint; Elevation : Shor'
18039:     +'tint; Azimuth : Smallint; SignLevel : Smallint; end');
18040:   CL.AddTypeS('TSatellites', 'array[1..12] of TSatellite');
18041: //TSatellites = array[1..MAX_SATS] of TSatellite;
18042:   TGPSSatEvent', 'Procedure ( Sender : TObject; NbSat, NbSatUse: Shortint; Sats : TSatellites )');
18043:   TGPSDatas', 'record Latitude : Double; Longitude : Double; Height'
18044:     + tAboveSea : Double; Speed : Double; UTCTime : TDateTime; Valid : Boolean; '
18045:     NbrSats : Shortint; NbrSatsUsed : Shortint; Course : Double; end');
18046:   CL.AddTypeS('TGPSDatasEvent', 'Procedure ( Sender : TObject; GPSDatas : TGPSDatas )');
18047:   CL.AddTypeS('TMsgGP', '( msgGP, msgGPGGA, msgGPGLL, msgGPGSV, msgGPRMA, msgGPRMC, msgGPZDA )');
18048:   CL.AddTypeS('TSpeedUnit', '( suKilometre, suMile, suNauticalMile )');
18049:   SIRegister_TGPSLink(CL);
18050:   SIRegister_TCustonGPS(CL);
18051:   SIRegister_TGPS(CL);
18052:   SIRegister_TGPSStoGPX(CL);
18053:   SIRegister_TGPSSpeed(CL);
18054:   SIRegister_TGPSSatellitesPosition(CL);

```

```

18055:   SIRegister_TGPSSatellitesReception(CL);
18056:   SIRegister_TGPSCompass(CL);
18057:   //CL.AddDelphiFunction('Procedure Register( )');
18058:   Function IndexMsgGP( StrMsgGP : String ) : TMsgGP';
18059:   Function StrCoordToAngle( Point : Char; Angle : String ) : Double';
18060:   Function StrTimeToTime( const Time : String ) : TDateTime';
18061:   Function StrToInteger( const Str : String ) : Integer';
18062:   Function StrToReal( const Str : String ) : Extended';
18063:   Function GPSRotatePoint( Angle : Double; Ct, Pt : TPoint ) : TPoint';
18064:   Procedure LoadRessource( RessourceName : String; ImageList : TImageList );
18065: end;
18066:
18067: procedure SIRegister_NMEA(CL: TPPascalCompiler);
18068: begin
18069:   NMEADataArray', 'array of string');
18070:   Procedure TrimNMEA( var S : string );
18071:   Procedure ExpandNMEA( var S : string );
18072:   Function ParseNMEA( S : string ) : NMEADataArray';
18073:   Function ChkValidNMEA( S : string ) : Boolean';
18074:   Function IdNMEA( S : string ) : string';
18075:   Function ChkSumNMEA( const S : string ) : string';
18076:   Function PosInDeg( const PosStr : string ) : Double';
18077:   Function DateTimeNMEA( const StrD, StrT : string ) : TDateTime';
18078:   Function SysClockSet( const StrD, StrT : string ) : Boolean';
18079:   function Ticks2Secs(Ticks : LongInt) : LongInt';
18080:   function Secs2Ticks(Secs : LongInt) : LongInt';
18081:   function MSecs2Ticks(MSecs : LongInt) : LongInt';
18082: end;
18083:
18084: function TRestRequest_createStringStreamFromStringList(strings: TStringList): TStringStream;
18085:
18086: {A simple Oscilloscope using TWaveIn class.
18087: More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
18088: uses
18089:   Forms,
18090:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
18091:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
18092:   uColorFunctions in 'uColorFunctions.pas',
18093:   AMixer in 'AMixer.pas',
18094:   uSettings in 'uSettings.pas',
18095:   UWavein4 in 'UWavein4.pas',
18096:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
18097:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
18098:
18099: Functions_max hex in the box maxbox
18100: functionslist.txt
18101: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98/100/101
18102:
18103: ****
18104: Procedure
18105: PROCEDURE SIZE 8396 8289 8242 7507 7401 6792 6310 5971 4438 3797 3600
18106: Procedure *****Now the Procedure list*****
18107: Procedure ( ACol, ARow : Integer; Items : TStrings)
18108: Procedure ( Agg : TAggregate)
18109: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
18110: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
18111: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
18112: Procedure ( ASender : TObject; const ABytes : Integer)
18113: Procedure ( ASender : TObject; VStream : TStream)
18114: Procedure ( AThread : TIdThread)
18115: Procedure ( AWebModule : TComponent)
18116: Procedure ( Column : TColumn)
18117: Procedure ( const AUsername : String; const APASSWORD : String; AAAuthenticationResult : Boolean)
18118: Procedure ( const iStart : integer; const sText : string)
18119: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
18120: Procedure ( Database : TDatabase; LoginParams : TStrings)
18121: Procedure ( DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:TReconcileAction)
18122: Procedure ( DATASET : TDATASET)
18123: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
18124: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
18125: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
18126: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
18127: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
18128: Procedure ( Done : Integer)
18129: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
18130: Procedure ( HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
18131: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
18132: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
18133: Procedure ( HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
18134: Procedure ( HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
18135: Procedure ( Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
18136: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
18137: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
18138: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
18139: Procedure ( SENDER : TFIELD; const TEXT : String)
18140: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : Boolean)
18141: Procedure ( Sender : TIdTelnet; const Buffer : String)
18142: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)

```

```

18143: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
18144: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18145: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
18146: Procedure ( Sender : Tobject; ACol, ARow : Longint; const Value : string)
18147: Procedure ( Sender : Tobject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
18148: Procedure ( Sender : Tobject; ACol, ARow : Longint; var CanSelect : Boolean)
18149: Procedure ( Sender : Tobject; ACol, ARow : Longint; var Value : string)
18150: Procedure ( Sender : Tobject; Button : TMPBtnType)
18151: Procedure ( Sender : Tobject; Button : TMPBtnType; var DoDefault : Boolean)
18152: Procedure ( Sender : Tobject; Button : TUDBtnType)
18153: Procedure ( Sender : Tobject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
18154: Procedure ( Sender : Tobject; ClientSocket : TServerClientWinSocket; var SocketThread : TServerClientThread)
18155: Procedure ( Sender : Tobject; Column : TListColumn)
18156: Procedure ( Sender : Tobject; Column : TListColumn; Point : TPoint)
18157: Procedure ( Sender : Tobject; Connecting : Boolean)
18158: Procedure ( Sender:TOObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var DoneDraw:Bool
18159: Procedure ( Sender:TOObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
18160: Procedure ( Sender : Tobject; const Rect : TRect; Field : TField; State : TGridDrawState)
18161: Procedure ( Sender : Tobject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
18162: Procedure ( Sender : Tobject; E : Exception; var Handled : Boolean)
18163: Procedure ( Sender : Tobject; FromIndex, ToIndex : Longint)
18164: Procedure ( Sender : Tobject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
18165: Procedure ( Sender : Tobject; Index : LongInt)
18166: Procedure ( Sender : Tobject; Item : TListItem)
18167: Procedure ( Sender : Tobject; Item : TListItem; Change : TItemChange)
18168: Procedure ( Sender : Tobject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
18169: Procedure ( Sender : Tobject; Item : TListItem; Selected : Boolean)
18170: Procedure ( Sender : Tobject; Item : TListItem; var AllowEdit : Boolean)
18171: Procedure ( Sender : Tobject; Item : TListItem; var S : string)
18172: Procedure ( Sender : Tobject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
18173: Procedure ( Sender : Tobject; ModalResult : TModalResult; var CanClose : Boolean)
18174: Procedure ( Sender : Tobject; Month : LongWord; var MonthBoldInfo : LongWord)
18175: Procedure ( Sender : Tobject; NewTab : Integer; var AllowChange : Boolean)
18176: Procedure ( Sender : Tobject; Node : TTreenode)
18177: Procedure ( Sender : Tobject; Node : TTreenode; var AllowChange : Boolean)
18178: Procedure ( Sender : Tobject; Node : TTreenode; var AllowCollapse : Boolean)
18179: Procedure ( Sender : Tobject; Node : TTreenode; var AllowEdit : Boolean)
18180: Procedure ( Sender : Tobject; Node : TTreenode; var AllowExpansion : Boolean)
18181: Procedure ( Sender : Tobject; Node : TTreenode; var S : string)
18182: Procedure ( Sender : Tobject; Node1, Node2 : TTreenode; Data : Integer; var Compare : Integer)
18183: Procedure ( Sender : Tobject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
18184: Procedure ( Sender : Tobject; Rect : TRect)
18185: Procedure ( Sender : Tobject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
18186: Procedure ( Sender:TOObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
18187: Procedure ( Sender : Tobject; Socket : TCustomWinSocket)
18188: Procedure ( Sender:TOObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
18189: Procedure ( Sender : Tobject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
18190: Procedure ( Sender : Tobject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
18191: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
18192: Procedure ( Sender : Tobject; StartPos, EndPos : Integer; var AllowChange : Boolean)
18193: Procedure ( Sender : Tobject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
18194: Procedure ( Sender : Tobject; TabIndex : Integer; var ImageIndex : Integer)
18195: Procedure ( Sender : Tobject; Thread : TServerClientThread)
18196: Procedure ( Sender : Tobject; TickCount : Cardinal; var Reset : Boolean)
18197: Procedure ( Sender : Tobject; Username, Password : string)
18198: Procedure ( Sender : Tobject; var AllowChange : Boolean)
18199: Procedure ( Sender : Tobject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
18200: Procedure ( Sender : Tobject; var Caption : string; var Alignment : THMLCaptionAlignment)
18201: Procedure ( Sender : Tobject; var Continue : Boolean)
18202: Procedure ( Sender:TOObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMETHOD:TIdHTTPMethod)
18203: Procedure ( Sender : Tobject; var Username : string)
18204: Procedure ( Sender : Tobject; Wnd : HWND)
18205: Procedure ( Sender : TToolbar; Button : TToolbutton)
18206: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
18207: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
18208: Procedure ( Sender : TToolBar; Index : Integer; var Allow : Boolean)
18209: Procedure ( Sender : TToolBar; Index : Integer; var Button : TToolbutton)
18210: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
18211: Procedure ( StatusBar : TStatusbar; Panel : TStatusPanel; const Rect : TRect)
18212: Procedure ( var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
18213: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
18214: procedure (Sender: TObject)
18215: procedure (Sender: TObject; var Done: Boolean)
18216: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
18217: procedure _T(Name: tbtString; v: Variant);
18218: Procedure AbandonSignalHandler( RtlSigNum : Integer)
18219: Procedure Abort
18220: Procedure About1Click( Sender : TObject)
18221: Procedure Accept( Socket : TSocket)
18222: Procedure AESSymmetricExecute(const plaintext, ciphertext, password: string)
18223: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
18224: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
18225: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
18226: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
18227: Procedure Add( Addend1, Addend2 : TMyBigInt)
18228: Procedure ADD( const AKEY, AVALUE : VARIANT)
18229: Procedure Add( const Key : string; Value : Integer)
18230: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)

```

```

18231: Procedure ADD( FIELD : TFIELD)
18232: Procedure ADD( ITEM : TMENUITEM)
18233: Procedure ADD( POPUP : TPOPUPMENU)
18234: Procedure AddCharacters( xCharacters : TCharSet)
18235: Procedure AddDriver( const Name : string; List : TStrings)
18236: Procedure AddImages( Value : TCustomImageList)
18237: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
18238: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
18239: Procedure AddLoader( Loader : TBitmapLoader)
18240: Procedure ADDPARAM( VALUE : TPARAM)
18241: Procedure AddPassword( const Password : string)
18242: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
18243: Procedure AddState( oState : TniRegularExpressionState)
18244: Procedure AddStrings( Strings : TStrings);
18245: procedure AddStrings(Strings: TStrings);
18246: Procedure AddStrings1( Strings : TWideStrings);
18247: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
18248: Procedure AddToRecentDocs( const Filename : string)
18249: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
18250: Procedure AllFunctionsList1Click( Sender : TObject)
18251: procedure AllObjectsList1Click(Sender: TObject);
18252: Procedure Allocate( AAllocateBytes : Integer)
18253: procedure AllResourceList1Click(Sender: TObject);
18254: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
18255: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
18256: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
18257: Procedure AnsiFree( var s : AnsiString)
18258: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
18259: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
18260: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
18261: Procedure Ansistring_to_stream( const Value : ansistring; Destin : TStream)
18262: Procedure AntiFreeze;
18263: Procedure APPEND
18264: Procedure Append( const S : WideString)
18265: procedure Append(S: string);
18266: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
18267: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
18268: Procedure AppendChunk( Val : OleVariant)
18269: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
18270: Procedure AppendStr( var Dest : string; S : string)
18271: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALen : Integer)
18272: Procedure ApplyRange
18273: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18274: Procedure Arrange( Code : TListArrangement)
18275: procedure Assert(expr : Boolean; const msg: string);
18276: procedure Assert2(expr : Boolean; const msg: string);
18277: Procedure Assign( Alist : TCustomBucketList)
18278: Procedure Assign( Other : TObject)
18279: Procedure Assign( Source : TDragObject)
18280: Procedure Assign( Source : TPersistent)
18281: Procedure Assign(Source: TPersistent)
18282: procedure Assign2(mystring, mypath: string);
18283: Procedure AssignCurValues( Source : TDataSet);
18284: Procedure AssignCurValues1( const CurValues : Variant);
18285: Procedure ASSIGNFIELD( FIELD : TFIELD)
18286: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
18287: Procedure AssignFile(var F: Text; FileName: string)
18288: procedure AssignFile(var F: TextFile; FileName: string)
18289: procedure AssignFileRead(var mystring, myfilename: string);
18290: procedure AssignFileWrite(mystring, myfilename: string);
18291: Procedure AssignTo( Other : TObject)
18292: Procedure AssignValues( Value : TParameters)
18293: Procedure ASSIGNVALUES( VALUE : TPARAMS)
18294: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
18295: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
18296: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
18297: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
18298: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18299: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
18300: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
18301: Procedure Bcddivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
18302: Procedure Bcddivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
18303: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
18304: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
18305: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
18306: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
18307: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18308: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
18309: procedure Beep
18310: Procedure BeepOk
18311: Procedure BeepQuestion
18312: Procedure BeepHand
18313: Procedure BeepExclamation
18314: Procedure BeepAsterisk
18315: Procedure BeepInformation
18316: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
18317: Procedure BeginLayout
18318: Procedure BeginTimer( const Delay, Resolution : Cardinal)
18319: Procedure BeginUpdate

```

```

18320: procedure BeginUpdate;
18321: procedure BigScreen1Click(Sender: TObject);
18322: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
18323: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
18324: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
18325: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
18326: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
18327: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord);
18328: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
18329: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
18330: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
18331: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
18332: Procedure BreakPointMenuClick( Sender : TObject);
18333: procedure BRINGTOFRONT;
18334: procedure BringToFront;
18335: Procedure btnBackClick( Sender : TObject);
18336: Procedure btnBrowseClick( Sender : TObject);
18337: Procedure BtnClick( Index : TNavigateBtn);
18338: Procedure btnLargeIconsClick( Sender : TObject);
18339: Procedure BuildAndSendRequest( AURI : TIdURI);
18340: Procedure BuildCache;
18341: Procedure BurnMemory( var Buff, BuffLen : integer);
18342: Procedure BurnMemoryStream( Destrukt : TMemoryStream);
18343: Procedure CalculateFirstSet;
18344: Procedure Cancel;
18345: procedure CancelDrag;
18346: Procedure CancelEdit;
18347: procedure CANCELHINT;
18348: Procedure CancelRange;
18349: Procedure CancelUpdates;
18350: Procedure CancelWriteBuffer;
18351: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool);
18352: Procedure Capture2(ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
18353: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool);
18354: procedure CaptureScreenFormat(vname: string; vextension: string);
18355: procedure CaptureScreenPNG(vname: string);
18356: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
18357: procedure CASCADE;
18358: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean);
18359: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
18360: Procedure cbPathClick( Sender : TObject);
18361: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState);
18362: Procedure cedebugAfterExecute( Sender : TPSScript);
18363: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal);
18364: Procedure cedebugCompile( Sender : TPSScript);
18365: Procedure cedebugExecute( Sender : TPSScript);
18366: Procedure cedebugIdle( Sender : TObject);
18367: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal);
18368: Procedure CenterHeight( const pc, pcParent : TControl);
18369: Procedure CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
18370: Procedure CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
18371: Procedure Change;
18372: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
18373: Procedure Changed;
18374: Procedure ChangeDir( const ADirName : string);
18375: Procedure ChangeDirUp;
18376: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState);
18377: Procedure ChangeLevelBy( Value : TChangeRange);
18378: Procedure ChDir(const s: string);
18379: Procedure Check(Status: Integer);
18380: Procedure CheckCommonControl( CC : Integer);
18381: Procedure CHECKFIELDNAME( const FIELDNAME : String);
18382: Procedure CHECKFIELDNAMES( const FIELDNAMES : String);
18383: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool);
18384: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean);
18385: Procedure CheckToken( T : Char);
18386: procedure CheckToken(t:char);
18387: Procedure CheckTokenSymbol( const S : string);
18388: procedure CheckTokenSymbol(s:string);
18389: Procedure CheckToolMenuDropdown( ToolButton : TToolButton);
18390: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18391: Procedure CIED65ToCIED50( var X, Y, Z : Extended);
18392: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
18393: procedure CipherFileClick(Sender: TObject);
18394: Procedure Clear;
18395: Procedure Clear1Click( Sender : TObject);
18396: Procedure ClearColor( Color : TColor);
18397: Procedure CLEARITEM( AITEM : TMENUITEM);
18398: Procedure ClearMapping;
18399: Procedure ClearSelection( KeepPrimary : Boolean);
18400: Procedure ClearWriteBuffer;
18401: Procedure Click;
18402: Procedure Close;
18403: Procedure Close1Click( Sender : TObject);
18404: Procedure CloseDatabase( Database : TDatabase);
18405: Procedure CloseDataSets;
18406: Procedure CloseDialog;
18407: Procedure CloseFile(var F: Text);
18408: Procedure Closure;

```

```

18409: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18410: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18411: Procedure CodeCompletionList1Click( Sender : TObject)
18412: Procedure ColEnter
18413: Procedure Collapse
18414: Procedure Collapse( Recurse : Boolean)
18415: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
18416: Procedure CommaSeparatedToStringList( Alist : TStringList; const Value : string)
18417: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
18418: Procedure Compile1Click( Sender : TObject)
18419: procedure ComponentCount1Click(Sender: TObject);
18420: Procedure Compress(azipfolder, azipfile: string)
18421: Procedure DeCompress(azipfolder, azipfile: string)
18422: Procedure XZip(azipfolder, azipfile: string)
18423: Procedure XUnZip(azipfolder, azipfile: string)
18424: Procedure Connect( const ATimeout : Integer)
18425: Procedure Connect( Socket : TSocket)
18426: procedure Console1Click(Sender: TObject);
18427: Procedure Continue
18428: Procedure ContinueCount( var Counter : TJclCounter)
18429: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
18430: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
18431: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
18432: Procedure ConvertImage(vsource, vdestination: string);
18433: // Ex. ConvertImage(Exepath+'my233.bmp',Exepath+'mypng111.png')
18434: Procedure ConvertBitmap(vsource, vdestination: string);
18435: Procedure ConvertToGray(Cnv: TCanvas);
18436: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
18437: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
18438: Procedure Copyl( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
18439: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
18440: Procedure CopyBytesToHostWord( const ASource : TIddBytes; const ASourceIndex : Integer; var VDest : Word)
18441: Procedure CopyFrom( mbCopy : TMyBigInt)
18442: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
18443: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
18444: Procedure CopyTIDByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array of Byte; const ADestIndex : Integer; const ALength : Integer)
18445: Procedure CopyTidBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const ALen:Int)
18446: Procedure CopyTidCardinal( const ASource : Cardinal; var VDest : TIddBytes; const ADestIndex : Integer)
18447: Procedure CopyTidInt64( const ASource : Int64; var VDest : TIddBytes; const ADestIndex : Integer)
18448: Procedure CopyTidIPv6Address(const ASource:TIdI Pv6Address; var VDest: TIddBytes; const ADestIndex : Integer)
18449: Procedure CopyTidLongWord( const ASource : LongWord; var VDest : TIddBytes; const ADestIndex : Integer)
18450: Procedure CopyTidNetworkLongWord( const ASource : LongWord; var VDest : TIddBytes; const ADestIndex : Integer)
18451: Procedure CopyTidNetworkWord( const ASource : Word; var VDest : TIddBytes; const ADestIndex : Integer)
18452: Procedure CopyTidString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
18453: Procedure CopyTidWord( const ASource : Word; var VDest : TIddBytes; const ADestIndex : Integer)
18454: Procedure CopyToClipboard
18455: Procedure CountParts
18456: Procedure CreateDataSet
18457: Procedure CreateEmptyFile( const FileName : string)
18458: Procedure CreateFileFromString( const FileName, Data : string)
18459: Procedure CreateFromDelta( Source : TPacketDataSet)
18460: procedure CREATEHANDLE
18461: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
18462: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
18463: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
18464: Procedure CreateTable
18465: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
18466: procedure CSyntax1Click(Sender: TObject);
18467: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
18468: Procedure CURSORPOSCHANGED
18469: procedure CutFirstDirectory(var S: String)
18470: Procedure DataBaseError(const Message: string)
18471: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
18472: procedure DateTimeToString(var Result : string; const Format : string; DateTime: TDateTime)
18473: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
18474: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
18475: Procedure DBIErrror(errorCode: Integer)
18476: Procedure DebugOutput( const AText : string)
18477: procedure DebugIn(DebugLOGFILE: string; Event_message: string);
18478: Procedure DebugRun1Click( Sender : TObject)
18479: procedure Dec;
18480: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
18481: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
18482: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
18483: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
18484: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,Aday,AHour,AMin,ASec,AMillSec:Word)
18485: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
18486: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
18487: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
18488: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
18489: Procedure Decompile1Click( Sender : TObject)
18490: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
18491: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
18492: Procedure DeferLayout
18493: Procedure defFilereread
18494: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)

```

```

18495: Procedure DelayMicroseconds( const MicroSeconds : Integer )
18496: Procedure Delete
18497: Procedure Delete( const Afilename : string )
18498: Procedure Delete( const Index : Integer )
18499: Procedure DELETE( INDEX : INTEGER )
18500: Procedure Delete( Index : LongInt )
18501: Procedure Delete( Node : TTreeNode )
18502: procedure Delete(var s: AnyString; ifrom, ictount: Longint );
18503: Procedure DeleteAlias( const Name : string )
18504: Procedure DeleteDriver( const Name : string )
18505: Procedure DeleteIndex( const Name : string )
18506: Procedure DeleteKey( const Section, Ident : String )
18507: Procedure DeleteRecords
18508: Procedure DeleteRecords( AffectRecords : TAffectRecords )
18509: Procedure DeleteString( var pStr : String; const pDelStr : string )
18510: Procedure DeleteTable
18511: procedure DelphiSiteClick(Sender: TObject );
18512: Procedure Deselect
18513: Procedure Deselect( Node : TTreeNode )
18514: procedure DestroyComponents
18515: Procedure DestroyHandle
18516: Procedure Diff( var X : array of Double )
18517: procedure Diff(var X: array of Double);
18518: Procedure DirCreate( const DirectoryName : String )';
18519: procedure DISABLEALIGN
18520: Procedure DisableConstraints
18521: Procedure Disconnect
18522: Procedure Disconnect( Socket : TSocket )
18523: Procedure Dispose
18524: procedure Dispose(P: PChar)
18525: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word )
18526: Procedure DoKey( Key : TDBCtrlGridKey )
18527: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView );
18528: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView );
18529: Procedure Dormant
18530: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd );
18531: Procedure DoubleToBytes( Value : Double; Bytes : array of byte )
18532: Procedure DoubleToComp( Value : Double; var Result : Comp )
18533: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
18534: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean );
18535: procedure Draw(X, Y: Integer; Graphic: TGraphic );
18536: Procedure Drawl(Canvas:TCanvas; X, Y,
    Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool );
18537: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer )
18538: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean )
18539: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer )
18540: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState )
18541: procedure DrawFocusRect(const Rect: TRect);
18542: Procedure DrawHIBToTBitmap( HDIB : THandle; Bitmap : TBitmap )
18543: Procedure DRAWMENUTEME( MENUTITEM : TMENUTITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE )
18544: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean );
18545: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
    TDrawingStyle; AImageType : TImageType; Enabled : Boolean );
18546: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
18547: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect )
18548: Procedure DropConnections
18549: Procedure DropDown
18550: Procedure DumpDescription( oStrings : TStrings )
18551: Procedure DumpStateTable( oStrings : TStrings )
18552: Procedure EDIT
18553: Procedure EditButtonClick
18554: Procedure EditFontClick( Sender : TObject )
18555: procedure Ellipse(X1, Y1, X2, Y2: Integer );
18556: Procedure Ellipse1( const Rect : TRect );
18557: Procedure EMMS
18558: Procedure Encode( ADest : TStream )
18559: procedure ENDDRAG(DROP:BOOLEAN)
18560: Procedure EndEdit( Cancel : Boolean )
18561: Procedure EndTimer
18562: Procedure EndUpdate
18563: Procedure EraseSection( const Section : string )
18564: Procedure Error( const Ident : string )
18565: procedure Error(Ident:Integer)
18566: Procedure ErrorFmt( const Ident : string; const Args : array of const )
18567: Procedure ErrorStr( const Message : string )
18568: procedure ErrorStr(Message:String)
18569: Procedure Exchange( Index1, Index2 : Integer )
18570: procedure Exchange(Index1, Index2: Integer);
18571: Procedure Exec( FileName, Parameters, Directory : string )
18572: Procedure ExecProc
18573: Procedure ExecSQL( UpdateKind : TUpdateKind )
18574: Procedure Execute
18575: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant )
18576: Procedure ExecuteAndWait( FileName : string; Visibility : Integer )
18577: Procedure ExecuteCommand(executeFile, paramstring: string)
18578: Procedure ExecuteShell(executeFile, paramstring: string)
18579: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18580: Procedure ExitThread(ExitCode: Integer); stdcall;
18581: Procedure ExitProcess(ExitCode: Integer); stdcall;

```

```

18582: Procedure Expand( AUserName : String; AResults : TStrings )
18583: Procedure Expand( Recurse : Boolean )
18584: Procedure ExportClipboardClick( Sender : TObject )
18585: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress )
18586: Procedure ExtractContentFields( Strings : TStrings )
18587: Procedure ExtractCookieFields( Strings : TStrings )
18588: Procedure ExtractFields( Separators,WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings )
18589: Procedure ExtractHeaderFields( Separators,WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings )
18590: Procedure ExtractHTTPHeaderFields( Separators,WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings )
18591: Procedure ExtractQueryFields( Strings : TStrings )
18592: Procedure FastDegToGrad
18593: Procedure FastDegToRad
18594: Procedure FastGradToDeg
18595: Procedure FastGradToRad
18596: Procedure FastRadToDeg
18597: Procedure FastRadToGrad
18598: Procedure FileClose( Handle : Integer )
18599: Procedure FileClose( handle: integer )
18600: procedure FilesFromWildcard( Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool )
18601: Procedure FileStructure( AStructure : TIIdFTPDataStructure )
18602: Procedure FillByte2( var X : Byte ; count: integer; value: byte )
18603: Procedure FillBytes( var VBytes : TIddBytes; const ACount : Integer; const AValue : Byte )
18604: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte )
18605: Procedure FillChar2( var X : PChar ; count: integer; value: char )
18606: Procedure FillCharS( var p : string; count: integer; value: char ); //fix3.8
18607: Procedure FillIPList
18608: procedure FillRect( const Rect: TRect );
18609: Procedure FillTStrings( AStrings : TStrings )
18610: Procedure FilterOnBookmarks( Bookmarks : array of const )
18611: procedure FinalizePackage( Module: HMODULE )
18612: procedure FindClose;
18613: procedure FindClose2( var F : TSearchRec )
18614: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent );
18615: Procedure FindMatches1( const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent );
18616: Procedure FindNearest( const KeyValues : array of const )
18617: Procedure FinishContext
18618: Procedure FIRST
18619: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float )
18620: Procedure FloatToDecimal( var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int );
18621: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle )
18622: Procedure FlushSchemaCache( const TableName : string )
18623: procedure FmtStr( var Result : string; const Format: string; const Args: array of const )
18624: procedure FOCUSCONTROL( CONTROL:TWINCONTROL )
18625: Procedure Form1Close( Sender : TObject; var Action : TCloseAction )
18626: Procedure FormActivate( Sender : TObject )
18627: procedure FormatLn( const format: String; const args: array of const ); //alias
18628: Procedure FormClose( Sender : TObject; var Action : TCloseAction )
18629: Procedure FormCreate( Sender : TObject )
18630: Procedure FormDestroy( Sender : TObject )
18631: Procedure FormKeyPress( Sender : TObject; var Key : Char )
18632: procedure FormOutput1Click(Sender: TObject);
18633: Procedure FormToHTML( Form : TForm; Path : string )
18634: procedure FrameRect( const Rect: TRect );
18635: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
18636: Procedure NotebookHandlesNeeded( Notebook : TNotebook )
18637: Procedure Free( Buffer : TRecordBuffer )
18638: Procedure Free( Buffer : TValueBuffer )
18639: Procedure Free;
18640: Procedure FreeAndNil( var Obj: TObject )
18641: Procedure FreeImage
18642: procedure FreeMem(P: PChar; Size: Integer)
18643: Procedure FreeTreeData( Tree : TUpdateTree )
18644: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer )
18645: Procedure FullCollapse
18646: Procedure FullExpand
18647: Procedure GenerateDB(sl: TStrings; var DPB: string; var DPBLength: Short); //InterBase
18648: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
18649: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML )
18650: Procedure Get1( AURL : string; const AResponseContent : TStream );
18651: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean );
18652: Procedure Get2( const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean );
18653: Procedure GetAliasNames( List : TStrings )
18654: Procedure GetAliasParams( const AliasName : string; List : TStrings )
18655: Procedure GetApplicationsRunning( Strings : TStrings )
18656: Procedure getBox(aURL, extension: string);
18657: Procedure GetCommandTypes( List : TWideStrings )
18658: Procedure GetConfigParams( const Path, Section : string; List : TStrings )
18659: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean )
18660: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray )
18661: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray )
18662: Procedure GetDatabaseNames( List : TStrings )
18663: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant )
18664: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD )
18665: Procedure GetDir(d: byte; var s: string)
18666: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean )
18667: Procedure GetDriverNames( List : TStrings )
18668: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean )

```

```

18669: Procedure GetDriverParams( const DriverName : string; List : TStrings)
18670: Procedure GetEmails1Click( Sender : TObject)
18671: Procedure getEnvironmentInfo;
18672: Function getEnvironmentString: string;
18673: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
18674: Procedure GetFieldNames( const TableName : string; List : TStrings)
18675: Procedure GetFieldNames( const TableName : string; List : TStrings);
18676: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
18677: Procedure GETFIELDNAMES( LIST : TSTRINGS)
18678: Procedure GetFieldNames1( const TableName : string; List : TStrings);
18679: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
18680: Procedure GetFieldNames2( const TableName : WideString; SchemaName : WideString; List : TWideStrings);
18681: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
18682: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
18683: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
18684: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
18685: Procedure GetFormatSettings
18686: Procedure GetFromDIB( var DIB : TBitmapInfo)
18687: Procedure GetFromHDIb( HDIB : HBitmap)
18688: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
18689: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
18690: Procedure GetIcon( Index : Integer; Image : TIcon);
18691: Procedure GetIconl(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
18692: Procedure GetIndexInfo( IndexName : string)
18693: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
18694: Procedure GetIndexNames( List : TStrings)
18695: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
18696: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
18697: Procedure GetIndexNames4( const TableName : string; List : TStrings);
18698: Procedure GetInternalResponse
18699: Procedure GETITEMNAMES( LIST : TSTRINGS)
18700: procedure GetMem(P: PChar; Size: Integer)
18701: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
18702: procedure GetPackageDescription(ModuleName: PChar): string)
18703: Procedure GetPackageNames( List : TStrings);
18704: Procedure GetPackageName1( List : TWideStrings);
18705: Procedure GetParamList( List : TList; const ParamNames : WideString)
18706: Procedure GetProcedureNames( List : TStrings);
18707: Procedure GetProcedureNames( List : TWideStrings);
18708: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
18709: Procedure GetProcedureNames1( List : TStrings);
18710: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
18711: Procedure GetProcedureNames3( List : TWideStrings);
18712: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
18713: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
18714: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
18715: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
18716: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
18717: Procedure GetProviderNames( Names : TWideStrings);
18718: Procedure GetProviderNames( Proc : TGetStrProc)
18719: Procedure GetProviderNames1( Names : TStrings);
18720: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
18721: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string); //no open image
18722: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const Data:string;aformat:string):TLinearBitmap;
18723: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
18724: Procedure GetSchemaNames( List : TStrings);
18725: Procedure GetSchemaNames1( List : TWideStrings);
18726: Procedure getScriptandRunAsk;
18727: Procedure getScriptandRun(ascript: string);
18728: Procedure getScript(ascript: string); //alias
18729: Procedure getWebScript(ascript: string); //alias
18730: Procedure GetSessionNames( List : TStrings)
18731: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
18732: Procedure GetStrings( List : TStrings)
18733: Procedure GetSystemTime; stdcall;
18734: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
18735: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
18736: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
18737: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
18738: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
18739: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
18740: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
18741: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
18742: Procedure GetVisibleWindows( List : TStrings)
18743: Procedure GoBegin
18744: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
18745: Procedure GotoCurrent( Table : TTable)
18746: procedure GotoEnd1Click(Sender: TObject);
18747: Procedure GotoNearest
18748: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
Direction: TGradientDirection)
18749: Procedure HandleException( E : Exception; var Handled : Boolean)
18750: procedure HANDLEMESSAGE
18751: procedure HandleNeeded;
18752: Procedure Head( AURL : string)
18753: Procedure Help( var AHelpContents : TStringList; ACommand : String)
18754: Procedure HexToBinary( Stream : TStream)
18755: procedure HexToBinary(Stream:TStream)

```

```

18756: Procedure HideDragImage
18757: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
18758: Procedure HideTraybar
18759: Procedure HideWindowForSeconds(secs: integer); //3 seconds
18760: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); //3 seconds
18761: Procedure HookOSExceptions
18762: Procedure HookSignal( RtlSigNum : Integer)
18763: Procedure HSLTORGBC( const H, S, L : Single; out R, G, B : Single);
18764: Procedure HTMLSyntax1Click( Sender : TObject)
18765: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
18766: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSEexec; x : TPSRuntimeClassImporter)
18767: Procedure ImportfromClipboard1Click( Sender : TObject)
18768: Procedure ImportfromClipboard2Click( Sender : TObject)
18769: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
18770: procedure Incb(var x: byte);
18771: Procedure Include1Click( Sender : TObject)
18772: Procedure IncludeOFF; //preprocessing
18773: Procedure IncludeON;
18774: procedure Info1Click(Sender: TObject);
18775: Procedure InitAltRecBuffers( CheckModified : Boolean)
18776: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
18777: Procedure InitContext( WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse )
18778: Procedure InitData( ASource : TDataSet)
18779: Procedure InitDelta( ADelta : TPacketDataSet);
18780: Procedure InitDelta( const ADelta : OleVariant);
18781: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
18782: Procedure Initialize
18783: procedure InitializePackage(Module: HMODULE)
18784: Procedure INITIACTION
18785: Procedure initHexArray(var hexn: THexArray); //THexArray', 'array[0..15] of char,'
18786: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
18787: Procedure InitModule( AModule : TComponent)
18788: Procedure InitStdConvs
18789: Procedure InitTreeData( Tree : TUpdateTree)
18790: Procedure INSERT
18791: Procedure Insert( Index : Integer; AClass : TClass)
18792: Procedure Insert( Index : Integer; AComponent : TComponent)
18793: Procedure Insert( Index : Integer; AObject : TObject)
18794: Procedure Insert( Index : Integer; const S : WideString)
18795: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
18796: Procedure Insert(Index: Integer; const S: string);
18797: procedure Insert(Index: Integer; S: string);
18798: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
18799: procedure InsertComponent(AComponent:TComponent)
18800: procedure InsertControl(AControl: TControl);
18801: Procedure InsertIcon( Index : Integer; Image : TIcon)
18802: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
18803: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
18804: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
18805: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
18806: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
18807: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
18808: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
18809: Procedure InternalBeforeResolve( Tree : TUpdateTree)
18810: Procedure InvalidateModuleCache
18811: Procedure InvalidateTitles
18812: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
18813: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
18814: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
ABaseDate:TDateTime)
18815: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
18816: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
18817: procedure JavaSyntax1Click(Sender: TObject);
18818: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
18819: Procedure KillDataChannel
18820: Procedure Largefont1Click( Sender : TObject)
18821: Procedure LAST
18822: Procedure LaunchCpl( FileName : string)
18823: Procedure Launch( const Afile : string)
18824: Procedure LaunchFile( const Afile : string)
18825: Procedure LetfileList(FileList: TStringlist; apath: string);
18826: Procedure lineToNumber( xmemo : String; met : boolean)
18827: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
DefaultDraw:Bool)
18828: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
: TCustomDrawState; var DefaultDraw : Boolean)
18829: Procedure ListViewData( Sender : TObject; Item : TListItem)
18830: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
: TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
18831: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
18832: Procedure ListViewDblClick( Sender : TObject)
18833: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18834: Procedure ListDLLExports(const FileName: string; List: TStrings);
18835: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
18836: procedure LoadBytecode1Click(Sender: TObject);
18837: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
18838: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
18839: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
18840: Procedure LoadFromFile( AFileName : string)

```

```

18841: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
18842: Procedure LoadFromFile( const FileName : string)
18843: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
18844: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
18845: Procedure LoadFromFile( const FileName : WideString)
18846: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18847: Procedure LoadFromFile(const AFileName: string)
18848: procedure LoadFromFile(fileName: string);
18849: procedure LoadFromFile(fileName:String)
18850: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
18851: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
18852: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
18853: Procedure LoadFromStream( const Stream : TStream)
18854: Procedure LoadFromStream( S : TStream)
18855: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18856: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18857: Procedure LoadFromStream( Stream : TStream)
18858: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
18859: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
18860: procedure LoadFromStream(Stream: TStream);
18861: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
18862: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
18863: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
18864: Procedure LoadLastFileClick( Sender : TObject)
18865: { LoadIconToImage loads two icons from resource named NameRes,
18866:   into two image lists ALarge and ASmall}
18867: Procedure LoadIconToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
18868: Procedure LoadMemo
18869: Procedure LoadParamsFromIniFile( FFileName : WideString)
18870: Procedure Lock
18871: Procedure Login
18872: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
18873: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
18874: Procedure MakeCaseInsensitive
18875: Procedure MakeDeterministic( var bChanged : boolean)
18876: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
18877: // type TVolumeLevel = 0..127; , saveFilePath as C:\MyFile.wav
18878: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
18879: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
18880: Procedure SetComplexSoundElements(freqdtt,Phaseedt,AmpEdt,WaveGrp:integer);
18881: Procedure SetRectComplexFormatStr( const S : string)
18882: Procedure SetPolarComplexFormatStr( const S : string)
18883: Procedure AddComplexSoundObjectToList(newf,newp,newa,neww:integer; freqlist: TStrings);
18884: Procedure MakeVisible
18885: Procedure MakeVisible( PartialOK : Boolean)
18886: Procedure Manual1Click( Sender : TObject)
18887: Procedure MarkReachable
18888: Procedure maxBox; //shows the exe version data in a win box
18889: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
18890: Procedure Memo1Change( Sender : TObject)
18891: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var Action:TSynReplaceAction)
18892: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
18893: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
18894: procedure Memory1Click(Sender: TObject);
18895: Procedure MERGE( MENU : TMAINMENU)
18896: Procedure MergeChangeLog
18897: procedure MINIMIZE
18898: Procedure MinimizeMaxbox;
18899: procedure MyCopyFile(Namel,Name2:string);
18900: Procedure MkDir(const s: string)
18901: Procedure MakeDir(const s: string)');
18902: Procedure ChangeDir(const s: string)');
18903: Function makefile(const FileName: string): integer)';
18904: Procedure mnuPrintFont1Click( Sender : TObject)
18905: procedure ModalStarted
18906: Procedure Modified
18907: Procedure ModifyAlias( Name : string; List : TStrings)
18908: Procedure ModifyDriver( Name : string; List : TStrings)
18909: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
18910: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
18911: Procedure Move( CurIndex, NewIndex : Integer)
18912: procedure Move(CurIndex, NewIndex: Integer);
18913: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
18914: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
18915: Procedure moveCube( o : TMyLabel )
18916: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
18917: procedure MoveTo(X, Y: Integer);
18918: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
18919: Procedure MovePoint(var x,y:Extended; const angle:Extended);
18920: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
18921: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
18922: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
18923: Procedure mxButton(x,y,width,height,top,left,aHandle: integer);
18924: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
18925: procedure New(P: PChar)
18926: procedure New1Click(Sender: TObject);
18927: procedure NewInstanceClick(Sender: TObject);
18928: Procedure NEXT

```

```

18929: Procedure NextMonth
18930: Procedure Noop
18931: Procedure NormalizePath( var APath : string)
18932: procedure ObjectBinaryToText(Input, Output: TStream)
18933: procedure ObjectBinaryToText1(Input, Output: TStream; var OriginalFormat: TStreamOriginalFormat)
18934: procedure ObjectResourceToText1(Input, Output: TStream)
18935: procedure ObjectResourceToText1(Input, Output: TStream; var OriginalFormat: TStreamOriginalFormat)
18936: procedure ObjectTextToBinary1(Input, Output: TStream)
18937: procedure ObjectTextToBinary1( Input, Output: TStream; var OriginalFormat: TStreamOriginalFormat)
18938: procedure ObjectTextToResource1( Input, Output: TStream)
18939: procedure ObjectTextToResource1( Input, Output: TStream; var OriginalFormat: TStreamOriginalFormat)
18940: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
18941: Procedure Open( const UserID : WideString; const Password : WideString);
18942: Procedure Open;
18943: Procedure open1Click( Sender : TObject)
18944: Procedure OpenCdDrive
18945: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
18946: Procedure OpenCurrent
18947: Procedure OpenFile(vfilenamepath: string)
18948: Procedure OpenDirectory1Click( Sender : TObject)
18949: Procedure OpenDir(adir: string);
18950: Procedure OpenIndexFile( const IndexName : string)
18951: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const SchemaID:OleVariant;DataSet:TADODataSet)
18952: Procedure OpenWriteBuffer( const AThreshold : Integer)
18953: Procedure OptimizeMem
18954: Procedure Options1( AURL : string);
18955: Procedure OutputDebugString(lpOutputString : PChar)
18956: Procedure PackBuffer
18957: Procedure Paint
18958: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
18959: Procedure PaintToTBitmap( Target : TBitmap)
18960: Procedure PaletteChanged
18961: Procedure ParentBiDiModeChanged
18962: Procedure PARENTBIDI MODECHANGED( ACONTROL : TOBJECT)
18963: Procedure PasteFromClipboard;
18964: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
18965: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
18966: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
18967: Procedure PError( Text : string)
18968: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18969: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
18970: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
18971: procedure playmp3(mpath: string);
18972: Procedure PlayMP31Click( Sender : TObject)
18973: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
18974: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
18975: procedure PolyBezier(const Points: array of TPoint);
18976: procedure PolyBezierTo(const Points: array of TPoint);
18977: procedure Polygon(const Points: array of TPoint);
18978: procedure Polyline(const Points: array of TPoint);
18979: Procedure Pop
18980: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU; GROUPS: array of INT; var WIDTHS : array of LONGINT)
18981: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
18982: Procedure POPUP( X, Y : INTEGER)
18983: Procedure PopupURL(URL : WideString);
18984: Procedure POST
18985: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
18986: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
18987: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
18988: Procedure PostUser( const Email, FirstName, LastName : WideString)
18989: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
18990: procedure Pred(X: int64);
18991: Procedure Prepare
18992: Procedure PrepareStatement
18993: Procedure PreProcessXML( AList : TStrings)
18994: Procedure PreventDestruction
18995: Procedure Print( const Caption : string)
18996: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
18997: procedure printf(const format: String; const args: array of const);
18998: Procedure PrintList(Value: TStringList);
18999: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle); //TBitmapStyle=(bsNormal,bsCentered,bsStretched)
19000: Procedure Printout1Click( Sender : TObject)
19001: Procedure ProcessHeaders
19002: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
19003: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
19004: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
19005: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
19006: Procedure ProcessMessagesOFF; //application.processmessages
19007: Procedure ProcessMessagesON;
19008: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
19009: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
19010: Procedure Proclist Size is: 3797 /1415
19011: Procedure procMessClick( Sender : TObject)
19012: Procedure PSScriptCompile( Sender : TPSScript)
19013: Procedure PSScriptExecute( Sender : TPSScript)
19014: Procedure PSScriptLine( Sender : TObject)
19015: Procedure Push( ABoundary : string)
19016: procedure PushItem(AItem: Pointer)

```

```

19017: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
19018: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
19019: procedure PutLinuxLines(const Value: string)
19020: Procedure Quit
19021: Procedure RaiseConversionError( const AText : string);
19022: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
19023: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
19024: procedure RaiseException(Ex: TIEException; Param: String);
19025: Procedure RaiseExceptionForLastCmdResult;
19026: procedure RaiseLastException;
19027: procedure RaiseException2;
19028: Procedure RaiseException3(const Msg: string);
19029: Procedure RaiseExcept(const Msg: string);
19030: Procedure RaiseLastOSError
19031: Procedure RaiseLastWin32;
19032: procedure RaiseLastWin32Error)
19033: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
19034: Procedure RandomFillStream( Stream : TMemoryStream)
19035: procedure randomize;
19036: Procedure Rasterize( TRasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
19037: Procedure RCS
19038: Procedure Read( Socket : TSocket)
19039: procedure ReadLn(var ast: string); //of inputquery
19040: Procedure ReadBlobData
19041: procedure ReadBuffer(Buffer:String;Count:LongInt)
19042: procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2,readonly:= false;
19043: Procedure ReadSection( const Section : string; Strings : TStrings)
19044: Procedure ReadSections( Strings : TStrings)
19045: Procedure ReadSections( Strings : TStrings);
19046: Procedure ReadSections1( const Section : string; Strings : TStrings);
19047: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
19048: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
19049: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
19050: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
19051: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
19052: Procedure Realign;
19053: procedure Rectangle(X1, Y1, X2, Y2: Integer);
19054: Procedure Rectangle1( const Rect : TRect);
19055: Procedure RectCopy( var Dest : TRect; const Source : TRect)
19056: Procedure RectFitToScreen( var R : TRect)
19057: Procedure RectGrow( var R : TRect; const Delta : Integer)
19058: Procedure RectGrowX( var R : TRect; const Delta : Integer)
19059: Procedure RectGrowY( var R : TRect; const Delta : Integer)
19060: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
19061: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
19062: Procedure RectNormalize( var R : TRect)
19063: // TFileCallbackProcedure = procedure(filename:string);
19064: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
19065: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
19066: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
19067: Procedure Refresh;
19068: Procedure RefreshData( Options : TFetchOptions)
19069: Procedure REFRESHLOOKUPLIST
19070: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
19071: Procedure RegExPathfinder2(Pathin, fileout, firstp, aregex, ext: string; asort, acopy: boolean);
19072: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIAuthENTICATIONCLASS)
19073: Procedure RegisterChanges( Value : TChangeLink)
19074: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
19075: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
19076: Procedure RegisterfileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
19077: Procedure ReInitialize( ADelay : Cardinal)
19078: procedure RELEASE
19079: Procedure Remove( const AByteCount : integer)
19080: Procedure REMOVE( FIELD : TFIELD)
19081: Procedure REMOVE( ITEM : TMENUITEM)
19082: Procedure REMOVE( POPUP : TPOPUPMENU)
19083: Procedure RemoveAllPasswords
19084: procedure RemoveComponent(AComponent:TComponent)
19085: Procedure RemoveDir( const ADirName : string)
19086: Procedure RemoveLambdaTransitions( var bChanged : boolean)
19087: Procedure REMOVEPARAM( VALUE : TPARAM)
19088: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
19089: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
19090: Procedure Rename( const ASourceFile, ADestFile : string)
19091: Procedure Rename( const FileName : string; Reload : Boolean)
19092: Procedure RenameTable( const NewTableName : string)
19093: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
19094: Procedure Replace1Click( Sender : TObject)
19095: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
19096: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime)
19097: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
19098: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
19099: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
19100: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
19101: Procedure Requery( Options : TExecuteOptions)
19102: Procedure Reset
19103: Procedure Reset1Click( Sender : TObject)
19104: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
19105: procedure ResourceExplore1Click(Sender: TObject);

```

```

19106: Procedure RestoreContents
19107: Procedure RestoreDefaults
19108: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
19109: Procedure RetrieveHeaders
19110: Procedure RevertRecord
19111: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
19112: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19113: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19114: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
19115: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
19116: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
19117: Procedure RleCompress2( Stream : TStream)
19118: Procedure RleDecompress2( Stream : TStream)
19119: Procedure RmDir(const S: string)
19120: Procedure Rollback
19121: Procedure Rollback( TransDesc : TTransactionDesc)
19122: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
19123: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
19124: Procedure RollbackTrans
19125: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
19126: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
19127: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
19128: Procedure RunD132Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
19129: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
19130: Procedure S_EBox( const AText : string)
19131: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var AddKey:int)
19132: Procedure S_IBox( const AText : string)
19133: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
19134: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
19135: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
19136: Procedure SampleVarianceAndMean
19137: ( const X : TDynFloatArray; var Variance, Mean : Float)
19138: Procedure Save2Click( Sender : TObject)
19139: Procedure Saveas3Click( Sender : TObject)
19140: Procedure Savebefore1Click( Sender : TObject)
19141: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
19142: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19143: Procedure SaveConfigFile
19144: Procedure SaveOutput1Click( Sender : TObject)
19145: procedure SaveScreenshotClick(Sender: TObject);
19146: Procedure SaveLn(pathname, content: string); //Saveln(exepath+'mysavelntest.txt', memo2.text);
19147: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
19148: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
19149: Procedure SaveToFile( AFileName : string)
19150: Procedure SAVETOFILE( const FILENAME : String)
19151: Procedure SaveToFile( const FileName : WideString)
19152: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
19153: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
19154: procedure SaveToFile(FileName: string);
19155: procedure SaveToFile(FileName:String)
19156: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
19157: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19158: Procedure SaveToStream( S : TStream)
19159: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19160: Procedure SaveToStream( Stream : TStream)
19161: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
19162: procedure SaveToStream(Stream: TStream);
19163: procedure SaveToStream(Stream:TStream)
19164: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
19165: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
19166: Procedure SaveToStreams( AStrings : TStrings; const MimeSeparator : Char)
19167: procedure Say(const sText: string)
19168: Procedure SBarcode1Click( Sender : TObject)
19169: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
19170: procedure ScriptExplorer1Click(Sender: TObject);
19171: Procedure Scroll( Distance : Integer)
19172: Procedure Scroll( DX, DY : Integer)
19173: procedure ScrollBy(DeltaX, DeltaY: Integer);
19174: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
19175: Procedure ScrollTabs( Delta : Integer)
19176: Procedure Search1Click( Sender : TObject)
19177: procedure SearchAndOpenDoc(vfilenamepath: string)
19178: procedure SearchAndOpenFile(vfilenamepath: string)
19179: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
19180: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19181: Procedure SearchNext1Click( Sender : TObject)
19182: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
19183: Procedure Select1( const Nodes : array of TTreeNode);
19184: Procedure Select2( Nodes : TList);
19185: Procedure SelectNext( Direction : Boolean)
19186: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
19187: Procedure SelfTestPEM //unit uPSI_OPEM
19188: Procedure Send( AMsg : TIdMessage)
19189: //config forst in const MAILINIFILE = 'maildef.ini';
19190: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','');
19191: Procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
19192: Procedure SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
19193: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)

```

```

19194: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
19195: Procedure SendResponse
19196: Procedure SendStream( AStream : TStream)
19197: procedure SendMCICmd(Cmd: string); !
19198: Procedure Set8087CW( NewCW : Word)
19199: Procedure SetAll( One, Two, Three, Four : Byte)
19200: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
19201: Procedure SetAppDispatcher( const AD dispatcher : TComponent)
19202: procedure SetArrayLength;
19203: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer); //2 dimension
19204: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19205: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer); //all init
19206: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
19207: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer');");
19208: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer');");
19209: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
19210: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
19211: Procedure SetAsHandle( Format : Word; Value : THandle)
19212: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
19213: procedure SetCaptureControl(Control: TControl);
19214: Procedure SetColumnAttributes
19215: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
19216: Procedure SetCustomHeader( const Name, Value : string)
19217: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
  FieldName:Widestring)
19218: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
19219: Procedure SetFocus
19220: procedure SetFocus; virtual;
19221: Procedure SetInitialState
19222: Procedure SetKey
19223: procedure SetLastError(ErrorCode: Integer)
19224: procedure SetLength;
19225: Procedure SetLineBreakStyle( var T : Text; Style : TTTextLineBreakStyle)
19226: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
19227: Procedure SetParams( ADataSet : TDataSet; UpdateKind : TUpdateKind);
19228: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
19229: Procedure SetParams1( UpdateKind : TUpdateKind);
19230: Procedure SetPassword( const Password : string)
19231: Procedure SetPointer( Ptr : Pointer; Size : Longint)
19232: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
19233: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
19234: Procedure SetProvider( Provider : TComponent)
19235: Procedure SetProxy( const Proxy : string)
19236: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
19237: Procedure SetRange( const StartValues, EndValues : array of const)
19238: Procedure SetRangeEnd
19239: Procedure SetRate( const aPercent, aYear : integer)
19240: procedure SetRate(const aPercent, aYear: integer)
19241: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19242: Procedure SetSafeCallExceptionMsg( Msg : String)
19243: procedure SETSELTEXTBUF(BUFFER:PCHAR)
19244: Procedure SetSize( AWidth, AHeight : Integer)
19245: procedure SetSize(NewSize:LongInt)
19246: procedure SetString(var s: string; buffer: PChar; len: Integer)
19247: Procedure SetStrings( List : TStrings)
19248: Procedure SetText( Text : PwideChar)
19249: procedure SetText(Text: Pchar);
19250: Procedure SetTextBuf( Buffer : PChar)
19251: procedure SETTEXTBUF(BUFFER:PCHAR)
19252: Procedure SetTick( Value : Integer)
19253: Procedure SetTimeout( ATimeOut : Integer)
19254: Procedure SetTraceEvent( Event : TDBXTraceEvent)
19255: Procedure SetUserName( const UserName : string)
19256: Procedure SetWallpaper( Path : string);
19257: procedure ShellStyle1Click(Sender: TObject);
19258: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
19259: Procedure ShowFileProperties( const FileName : string)
19260: Procedure ShowIncludelClick( Sender : TObject)
19261: Procedure ShowInterfaces1Click( Sender : TObject)
19262: Procedure ShowLastException1Click( Sender : TObject)
19263: Procedure ShowMessage( const Msg : string)
19264: Procedure ShowMessageBig(const aText : string);
19265: Procedure ShowMessageBig2(const aText : string; aaAutoSize: boolean);
19266: Procedure ShowMessageBig3(const aText : string; fsize: byte; aaAutoSize: boolean);
19267: Procedure MsgBig(const aText : string); //alias
19268: procedure showmessage(mytext: string);
19269: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
19270: procedure ShowMessageFmt(const Msg: string; Params: array of const))
19271: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
19272: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
19273: Procedure ShowSearchDialog( const Directory : string)
19274: Procedure ShowSpecChars1Click( Sender : TObject)
19275: Procedure ShowBitmap(bmap: TBitmap); //draw in a form!
19276: Procedure ShredFile( const FileName : string; Times : Integer)
19277: procedure Shuffle(vQ: TStringList);
19278: Procedure ShuffleList( var List : array of Integer; Count : Integer)
19279: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
19280: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
19281: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)

```

```

19282: Procedure Site( const ACommand : string)
19283: Procedure SkipEOL
19284: Procedure Sleep( ATime : cardinal)
19285: Procedure Sleep( milliseconds : Cardinal)
19286: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
19287: Procedure Slinenumbers1Click( Sender : TObject)
19288: Procedure Sort
19289: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
19290: procedure SoundAlarm; //beep seq
19291: procedure Speak(const sText: string) //async like voice
19292: procedure Speak2(const sText: string) //sync
19293: procedure Split(Str: string; SubStr: string; List: TStrings);
19294: Procedure SplitNameValuePair( const Line : string; var Name, Value : string)
19295: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
19296: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
19297: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
19298: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
19299: procedure SQLSyntax1Click(Sender: TObject);
19300: Procedure SRand( Seed : RNG_IntType)
19301: Procedure Start
19302: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
19303: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
19304: //Ex. StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
19305: Procedure StartTransaction( TransDesc : TTransactionDesc)
19306: Procedure Status( var AStatusList : TStringList)
19307: Procedure StatusBar1DblClick( Sender : TObject)
19308: Procedure StepInto1Click( Sender : TObject)
19309: Procedure StepIt
19310: Procedure StepOut1Click( Sender : TObject)
19311: Procedure Stop
19312: procedure stopmp3;
19313: procedure StartWeb(aurl: string);
19314: Procedure Str(aint: integer; astr: string); //of system
19315: Procedure StrDispose( Str : PChar)
19316: procedure StrDispose(Str: PChar)
19317: Procedure StrReplace(var Str: String; Old, New: String);
19318: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
19319: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
19320: Procedure StringToBytes( Value : String; Bytes : array of byte)
19321: procedure StrSet(c : Char; I : Integer; var s : String);
19322: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19323: Procedure StructureMount( APath : String)
19324: procedure STYLECHANGED(SENDER:TOBJECT)
19325: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
19326: procedure Succ(X: int64);
19327: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
19328: procedure SwapChar(var X,Y: char); //swapX follows
19329: Procedure SwapFloats( var X, Y : Float)
19330: procedure SwapGrid(grd: TStringGrid);
19331: Procedure SwapOrd( var I, J : Byte);
19332: Procedure SwapOrd( var X, Y : Integer)
19333: Procedure SwapOrd1( var I, J : Shortint);
19334: Procedure SwapOrd2( var I, J : Smallint);
19335: Procedure SwapOrd3( var I, J : Word);
19336: Procedure SwapOrd4( var I, J : Integer);
19337: Procedure SwapOrd5( var I, J : Cardinal);
19338: Procedure SwapOrd6( var I, J : Int64);
19339: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
19340: Procedure Synchronize( Method : TMethod);
19341: procedure SyntaxCheck1Click(Sender: TObject);
19342: Procedure SysFreeString(const S: WideString); stdcall;
19343: Procedure TakeOver( Other : TLinearBitmap)
19344: Procedure Talkln(const sText: string) //async voice
19345: procedure tbtn6resClick(Sender: TObject);
19346: Procedure tbtnUseCaseClick( Sender : TObject)
19347: procedure TerminalStyle1Click(Sender: TObject);
19348: Procedure Terminate
19349: Procedure texSyntax1Click( Sender : TObject)
19350: procedure TextOut(X, Y: Integer; Text: string);
19351: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
19352: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
19353: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
19354: Procedure TextStart
19355: procedure TILE
19356: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
19357: Procedure TitleClick( Column : TColumn)
19358: Procedure ToDo
19359: procedure toolbtnTutorialClick(Sender: TObject);
19360: Procedure Trace1( AURL : string; const AResponseContent : TStream);
19361: Procedure TransferMode( ATransferMode : TIdFTPTTransferMode)
19362: Procedure Truncate
19363: procedure Tutorial101Click(Sender: TObject);
19364: procedure Tutorial10Statistics1Click(Sender: TObject);
19365: procedure Tutorial11Forms1Click(Sender: TObject);
19366: procedure Tutorial12SQL1Click(Sender: TObject);
19367: Procedure tutorial1Click( Sender : TObject)
19368: Procedure tutorial2Click( Sender : TObject)
19369: Procedure tutorial3Click( Sender : TObject)
19370: Procedure tutorial4Click( Sender : TObject)

```

```

19371: Procedure Tutorial5Click( Sender : TObject )
19372: procedure Tutorial6Click(Sender: TObject);
19373: procedure Tutorial91Click(Sender: TObject);
19374: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
19375: procedure UniqueString(var str: AnsiString)
19376: procedure UnloadLoadPackage(Module: HMODULE)
19377: Procedure Unlock
19378: Procedure UNMERGE( MENU : TMAINMENU)
19379: Procedure UnRegisterChanges( Value : TChangeLink)
19380: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
19381: Procedure UnregisterConversionType( const AType : TConvType)
19382: Procedure UnRegisterProvider( Prov : TCustomProvider)
19383: Procedure UPDATE
19384: Procedure UpdateBatch( AffectRecords : TAffectRecords)
19385: Procedure UPDATECURSORPOS
19386: Procedure UpdateFile
19387: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
19388: Procedure UpdateResponse( AResponse : TWebResponse)
19389: Procedure UpdateScrollBar
19390: Procedure UpdateView1Click( Sender : TObject)
19391: procedure Val(const s: string; var n, z: Integer)
19392: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
19393: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
19394: Procedure VariantAdd( const src : Variant; var dst : Variant)
19395: Procedure VariantAnd( const src : Variant; var dst : Variant)
19396: Procedure VariantArrayRedim( var V : Variant; High : Integer)
19397: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
19398: Procedure VariantClear( var V : Variant)
19399: Procedure VariantCpy( const src : Variant; var dst : Variant)
19400: Procedure VariantDiv( const src : Variant; var dst : Variant)
19401: Procedure VariantMod( const src : Variant; var dst : Variant)
19402: Procedure VariantMul( const src : Variant; var dst : Variant)
19403: Procedure VariantOr( const src : Variant; var dst : Variant)
19404: Procedure VariantPutElement( var V : Variant; const data : Variant; il : integer);
19405: Procedure VariantPutElement1( var V : Variant; const data : Variant; il, i2 : integer);
19406: Procedure VariantPutElement2( var V : Variant; const data : Variant; il, i2, i3 : integer);
19407: Procedure VariantPutElement3( var V : Variant; const data : Variant; il, i2, i3, i4 : integer);
19408: Procedure VariantPutElement4( var V : Variant; const data : Variant; il, i2, i3, i4, i5 : integer);
19409: Procedure VariantShl( const src : Variant; var dst : Variant)
19410: Procedure VariantShr( const src : Variant; var dst : Variant)
19411: Procedure VariantSub( const src : Variant; var dst : Variant)
19412: Procedure VariantXor( const src : Variant; var dst : Variant)
19413: Procedure VarCastError;
19414: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
19415: Procedure VarInvalidOp
19416: Procedure VarInvalidNullOp
19417: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
19418: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
19419: Procedure VarArrayCreateError
19420: Procedure VarResultCheck( AResult : HRESULT);
19421: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
19422: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
19423: Function VarTypeAsText( const AType : TVarType) : string
19424: procedure Voice(const sText: string) //async
19425: procedure Voice2(const sText: string) //sync
19426: Procedure WaitMilliseconds( AMSec : word)
19427: Procedure WaitMS( AMSec : word)');
19428: procedure WebCamPic(picname: string); //eg: c:\mypic.png
19429: Procedure WideAppend( var dst : WideString; const src : WideString)
19430: Procedure WideAssign( var dst : WideString; var src : WideString)
19431: Procedure WideDelete( var dst : WideString; index, count : Integer)
19432: Procedure WideFree( var s : WideString)
19433: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
19434: Procedure WideFromPChar( var dst : WideString; src : PChar)
19435: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
19436: Procedure WidestLength( var dst : WideString; len : Integer)
19437: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
19438: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
19439: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
19440: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19441: Procedure HttpGet(const Url: string; Stream:TStream);
19442: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
19443: Procedure WordWrap1Click( Sender : TObject)
19444: Procedure Write( const Aout : string)
19445: Procedure Write( Socket : TSocket)
19446: procedure Write(S: string);
19447: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
19448: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
19449: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
19450: procedure WriteBuffer(Buffer:String;Count:LongInt)
19451: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
19452: Procedure WriteChar( AValue : Char)
19453: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
19454: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
19455: Procedure WriteFloat( const Section, Name : string; Value : Double)
19456: Procedure WriteHeader( AHeader : TStrings)
19457: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
19458: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
19459: Procedure WriteLn( const AOut : string)

```

```

19460: procedure Writeln(s: string);
19461: Procedure WriteLog( const FileName, LogLine : string)
19462: Procedure WriteRFCReply( AReply : TIdRFCReply)
19463: Procedure WriteRFCStrings( AStrings : TStrings)
19464: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
19465: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
19466: Procedure WriteString( const Section, Ident, Value : String)
19467: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
19468: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
19469: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
19470: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19471: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19472: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)'')
19473: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
19474: procedure XMLSyntax1Click(Sender: TObject);
19475: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
19476: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
19477: Procedure ZeroFillStream( Stream : TMemoryStream)
19478: procedure XMLSyntax1Click(Sender: TObject);
19479: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
19480: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
19481: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
19482: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
19483: procedure(Sender, Source: TObject; X, Y: Integer)
19484: procedure(Sender, Target: TObject; X, Y: Integer)
19485: procedure(Sender: TObject; ASection, AWidth: Integer)
19486: procedure(Sender: TObject; ScrollCode: TScrollCode; var ScrollPos: Integer)
19487: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
19488: procedure(Sender: TObject; var Action: TCloseAction)
19489: procedure(Sender: TObject; var CanClose: Boolean)
19490: procedure(Sender: TObject; var Key: Char);
19491: ProcedureName ProcedureNames ProcedureParametersCursor @
19492:
19493: *****Now Constructors constructor *****
19494: Size is: 1248 1115 996 628 550 544 501 459 (381)
19495: Attach(VersionInfoData : Pointer; Size : Integer)
19496: constructor Create( ABuckets : TBucketListSizes)
19497: Create( ACallBackWnd : HWnd)
19498: Create( AClient : TCustomTaskDialog)
19499: Create( AClient : TIdTelnet)
19500: Create( ACollection : TCollection)
19501: Create( ACollection : TFavoriteLinkItems)
19502: Create( ACollection : TTaskDialogButtons)
19503: Create( AConnection : TIdCustomHTTP)
19504: Create( ACreateSuspended : Boolean)
19505: Create( ADataSet : TCustomSQLDataSet)
19506: CREATE( ADATASET : TDATASET)
19507: Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
19508: Create( AGrid : TCustomDBGrid)
19509: Create( AGrid : TStringGrid; AIndex : Longint)
19510: Create( AHHTTP : TIdCustomHTTP)
19511: Create( AListItems : TListItems)
19512: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19513: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19514: Create( AOwner : TCommonCalendar)
19515: Create( AOwner : TComponent)
19516: CREATE( AOWNER : TCOMPONENT)
19517: Create( AOwner : TCustomListView)
19518: Create( AOwner : TCustomOutline)
19519: Create( AOwner : TCustomRichEdit)
19520: Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
19521: Create( AOwner : TCustomTreeView)
19522: Create( AOwner : TIdUserManager)
19523: Create( AOwner : TListItems)
19524: Create(AOwner:TObj;Handle:hDBCICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
19525: CREATE( AOWNER : TPERSISTENT)
19526: Create( AOwner : TPersistent)
19527: Create( AOwner : TTable)
19528: Create( AOwner : TTreeNodes)
19529: Create( AOwner : TWinControl; const ClassName : string)
19530: Create( AParent : TIdCustomHTTP)
19531: Create( AParent : TUpdateTree; AResolver : TCustomResolver)
19532: Create( AProvider : TBaseProvider)
19533: Create( AProvider : TCustomProvider);
19534: Create( AProvider : TDataSetProvider)
19535: Create( ASocket : TCustomWinSocket; TimeOut : Longint)
19536: Create( ASocket : TSocket)
19537: Create( AStrings : TWideStrings)
19538: Create( AToolBar : Toolbar)
19539: Create( ATreeNodes : TTreeNodes)
19540: Create( Autofill : boolean)
19541: Create( AWebPageInfo : TAbstractWebPageInfo)
19542: Create( AWebRequest : TWebRequest)
19543: Create( Collection : TCollection)
19544: Create( Collection : TIdMessageParts; ABody : TStrings)
19545: Create( Collection : TIdMessageParts; const AFileName : TFileName)
19546: Create( Column : TColumn)
19547: Create( const AConvFamily : TConvFamily; const ADescription : string)
19548: Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)

```

```

19549: Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
  AFromCommonProc : TConversionProc)
19550: Create( const AInitialState : Boolean; const AManualReset : Boolean)
19551: Create( const ATabSet : TTabSet)
19552: Create( const Compensate : Boolean)
19553: Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
19554: Create( const FileName : string)
19555: Create( const FileName : string; FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
  : Int64; const SecAttr : PSecurityAttributes)
19556: Create( const FileName : string; FileMode : WordfmShareDenyWrite)
19557: Create( const MaskValue : string)
19558: Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
19559: Create( const Prefix : string)
19560: Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
19561: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19562: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19563: Create( CoolBar : TCoolBar)
19564: Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
19565: Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
19566: Create( DataSet : TDataSet; const
  Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
  DepFields : TBits; FieldMap : TFieldMap)
19567: Create( DBCtrlGrid : TDBCtrlGrid)
19568: Create( DSTableProducer : TDSTableProducer)
19569: Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
19570: Create( ErrorCode : DBIResult)
19571: Create( Field : TBlobField; Mode : TBlobStreamMode)
19572: Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
19573: Create( HeaderControl : TCustomHeaderControl)
19574: Create( HTTPRequest : TWebRequest)
19575: Create( iStart : integer; sText : string)
19576: Create( iValue : Integer)
19577: Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
19578: Create( MciErrNo : MCIERROr; const Msg : string)
19579: Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
19580: Create( Message : string; ErrorCode : DBResult)
19581: Create( Msg : string)
19582: Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
19583: Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
19584: Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
19585: Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
19586: Create(oSource:TniRegularExpressionState;oDestination:TniRegularExpressionState;xCharacts:TCharSet;bLambda:bool)
19587: Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
19588: Create( Owner : TCustomComboBoxEx)
19589: CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
19590: Create( Owner : TPersistent)
19591: Create( Params : TStrings)
19592: Create( Size : Cardinal)
19593: Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
19594: Create( StatusBar : TCustomStatusBar)
19595: Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
19596: Create(WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
19597: Create(AHandle:Integer)
19598: Create(AOwner: TComponent); virtual;
19599: Create(const AURI : string)
19600: Create(FileName:String;Mode:Word)
19601: Create(Instance:THandle;ResName:String;ResType:PChar)
19602: Create(Stream : TStream)
19603: Createel( ADataset : TDataset);
19604: Createel(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
  SecAttr:PSecurityAttributes);
19605: Createel( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
19606: Create2( Other : TObject);
19607: CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
19608: CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
19609: CreateFmt( MciErrNo : MCIERROr; const Msg : string; const Args : array of const)
19610: CreateFromId(InstanceId:THandle;ResId:Integer;ResType:PChar)
19611: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
19612: CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
19613: CreateRes( Ident : Integer);
19614: CreateRes( MciErrNo : MCIERROr; Ident : Integer)
19615: CreateRes( ResStringRec : PResStringRec);
19616: CreateResHelp( Ident : Integer; AHelpContext : Integer);
19617: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
19618: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
19619: CreateSize( AWidth, AHeight : Integer)
19620: Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
19621:
19622: -----
19623: unit uPSI_MathMax;
19624: -----
19625: CONSTS
19626: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
19627: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
19628: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
19629: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
19630: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
19631: CbrtPi: Float = 1.4645918875615232630201425272638; // CubeRoot(PI)
19632: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant

```

```

19633: PiJ: Float      = 3.1415926535897932384626433832795; // PI
19634: PI: Extended = 3.1415926535897932384626433832795;
19635: PiOn2: Float    = 1.5707963267948966192313216916398; // PI / 2
19636: PiOn3: Float    = 1.0471975511965977461542144610932; // PI / 3
19637: PiOn4: Float    = 0.78539816339744830961566084581988; // PI / 4
19638: Sqrt2: Float    = 1.4142135623730950488016887242097; // Sqrt(2)
19639: Sqrt3: Float    = 1.7320508075688772935274463415059; // Sqrt(3)
19640: Sqrt5: Float    = 2.2360679774997896964091736687313; // Sqrt(5)
19641: Sqrt10: Float   = 3.162277660168379319988935444327; // Sqrt(10)
19642: SqrtPi: Float   = 1.7724538509055160272981674833411; // Sqrt(PI)
19643: Sqrt2Pi: Float  = 2.506628274631000502415765284811; // Sqrt(2 * PI)
19644: TwoPi: Float   = 6.283185307179586476925286766559; // 2 * PI
19645: ThreePi: Float = 9.4247779607693797153879301498385; // 3 * PI
19646: Ln2: Float     = 0.69314718055994530941723212145818; // Ln(2)
19647: Ln10: Float    = 2.3025850929940456840179914546844; // Ln(10)
19648: LnPi: Float   = 1.1447298858494001741434273513531; // Ln(PI)
19649: Log2J: Float   = 0.30102999566398119521373889472449; // Log10(2)
19650: Log3: Float    = 0.47712125471966243729502790325512; // Log10(3)
19651: LogPi: Float  = 0.4971498726941338543512682882909; // Log10(PI)
19652: LogE: Float   = 0.434294481903251827651128918911661; // Log10(E)
19653: E: Float       = 2.7182818284590452353602874713527; // Natural constant
19654: hLn2Pi: Float = 0.91893853320467274178032973640562; // Ln(2*PI)/2
19655: inv2Pi: Float = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
19656: TwoToPower63: Float = 9223372036854775808.0; // 2^63
19657: GoldenMean: Float = 1.618033988749894848204586834365638; // GoldenMean
19658: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
19659: RadCor : Double = 57.29577951308232; {number of degrees in a radian}
19660: StDelta : Extended = 0.00001; {delta for difference equations}
19661: StEpsilon : Extended = 0.00001; {epsilon for difference equations}
19662: StMaxIterations : Integer = 100; {max attempts for convergence}
19663:
19664: procedure SIRegister_StdConvs(CL: TPSCompiler);
19665: begin
19666:   MetersPerInch = 0.0254; // [1]
19667:   MetersPerFoot = MetersPerInch * 12;
19668:   MetersPerYard = MetersPerFoot * 3;
19669:   MetersPerMile = MetersPerFoot * 5280;
19670:   MetersPerNauticalMiles = 1852;
19671:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
19672:   MetersPerLightSecond = 2.99792458E8; // [5]
19673:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
19674:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
19675:   MetersPerCubit = 0.4572; // [6][7]
19676:   MetersPerFathom = MetersPerFoot * 6;
19677:   MetersPerFurlong = MetersPerYard * 220;
19678:   MetersPerHand = MetersPerInch * 4;
19679:   MetersPerPace = MetersPerInch * 30;
19680:   MetersPerRod = MetersPerFoot * 16.5;
19681:   MetersPerChain = MetersPerRod * 4;
19682:   MetersPerLink = MetersPerChain / 100;
19683:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
19684:   MetersPerPica = MetersPerPoint * 12;
19685:
19686:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
19687:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
19688:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
19689:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
19690:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
19691:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
19692:
19693:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
19694:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
19695:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
19696:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
19697:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
19698:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
19699:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
19700:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
19701:
19702:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
19703:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
19704:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
19705:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
19706:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
19707:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
19708:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
19709:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
19710:
19711:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
19712:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
19713:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
19714:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
19715:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
19716:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
19717:
19718:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
19719:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
19720:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
19721:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;

```

```

19722: CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
19723: CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
19724: CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
19725: CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
19726: CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
19727:
19728: GramsPerPound = 453.59237; // [1][7]
19729: GramsPerDrams = GramsPerPound / 256;
19730: GramsPerGrains = GramsPerPound / 7000;
19731: GramsPerTons = GramsPerPound * 2000;
19732: GramsPerLongTons = GramsPerPound * 2240;
19733: GramsPerOunces = GramsPerPound / 16;
19734: GramsPerStones = GramsPerPound * 14;
19735:
19736: MaxAngle 9223372036854775808.0;
19737: MaxTanH 5678.2617031470719747459655389854);
19738: MaxFactorial( 1754);
19739: MaxFloatingPoint(1.189731495357231765085759326628E+4932);
19740: MinFloatingPoint'(3.3621031431120935062626778173218E-4932);
19741: MaxTanH( 354.89135644669199842162284618659);
19742: MaxFactorial'LongInt'( 170);
19743: MaxFloatingPointD(1.797693134862315907729305190789E+308);
19744: MinFloatingPointD(2.2250738585072013830902327173324E-308);
19745: MaxTanh( 44.361419555836499802702855773323);
19746: MaxFactorial'LongInt'( 33);
19747: MaxFloatingPointS( 3.4028236692093846346337460743177E+38);
19748: MinFloatingPointS( 1.1754943508222875079687365372222E-38);
19749: PiExt( 3.1415926535897932384626433832795);
19750: RatioDegToRad( PiExt / 180.0);
19751: RatioGradToRad( PiExt / 200.0);
19752: RatioDegToGrad( 200.0 / 180.0);
19753: RatioGradToDeg( 180.0 / 200.0);
19754: Crc16PolynomCCITT'LongWord $1021);
19755: Crc16PolynomIBM'LongWord $8005);
19756: Crc16Bits'LongInt'( 16);
19757: Crc16Bytes'LongInt'( 2);
19758: Crc16HighBit'LongWord $8000);
19759: NotCrc16HighBit', 'LongWord $7FFF);
19760: Crc32PolynomIEEE', 'LongWord $04C11DB7);
19761: Crc32PolynomCastagnoli', 'LongWord $1EDC6F41);
19762: Crc32Koopman', 'LongWord $741B8CD7);
19763: Crc32Bits', 'LongInt'( 32);
19764: Crc32Bytes', 'LongInt'( 4);
19765: Crc32HighBit', 'LongWord $80000000);
19766: NotCrc32HighBit', 'LongWord $7FFFFFFF);
19767:
19768: MinByte      = Low(Byte);
19769: MaxByte      = High(Byte);
19770: MinWord      = Low(Word);
19771: MaxWord      = High(Word);
19772: MinShortInt = Low(ShortInt);
19773: MaxShortInt = High(ShortInt);
19774: MinSmallInt = Low(SmallInt);
19775: MaxSmallInt = High(SmallInt);
19776: MinLongWord = LongWord(Low(LongWord));
19777: MaxLongWord = LongWord(High(LongWord));
19778: MinLongInt = LongInt(Low(LongInt));
19779: MaxLongInt = LongInt(High(LongInt));
19780: MinInt64    = Int64(Low(Int64));
19781: MaxInt64    = Int64(High(Int64));
19782: MinInteger = Integer(Low(Integer));
19783: MaxInteger = Integer(High(Integer));
19784: MinCardinal = Cardinal(Low(Cardinal));
19785: MaxCardinal = Cardinal(High(Cardinal));
19786: MinNativeUInt = NativeUInt(Low(NativeUInt));
19787: MaxNativeUInt = NativeUInt(High(NativeUInt));
19788: MinNativeInt = NativeInt(Low(NativeInt));
19789: MaxNativeInt = NativeInt(High(NativeInt));
19790: Function CosH( const Z : Float) : Float;
19791: Function SinH( const Z : Float) : Float;
19792: Function TanH( const Z : Float) : Float;
19793:
19794: //*****from DMath.Dll Lib of types.inc in source\dmath_dll
19795: InvLn2      = 1.44269504088896340736; { 1/Ln(2) }
19796: InvLn10     = 0.43429448190325182765; { 1/Ln(10) }
19797: TwoPi       = 6.28318530717958647693; { 2*Pi }
19798: PiDiv2     = 1.57079632679489661923; { Pi/2 }
19799: SqrtPi     = 1.77245385090551602730; { Sqrt(Pi) }
19800: Sqrt2Pi    = 2.50662827463100050242; { Sqrt(2*Pi) }
19801: InvSqrt2Pi = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
19802: LnSqrt2Pi = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
19803: Ln2PiDiv2 = 0.91893853320467274178; { Ln(2*Pi)/2 }
19804: Sqrt2      = 1.41421356237309504880; { Sqrt(2) }
19805: Sqrt2Div2 = 0.70710678118654752440; { Sqrt(2)/2 }
19806: Gold       = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
19807: CGold      = 0.38196601125010515179; { 2 - GOLD }
19808: MachEp    = 2.220446049250313E-16; { 2^(-52) }
19809: MaxNum     = 1.797693134862315E+308; { 2^1024 }
19810: MinNum     = 2.225073858507202E-308; { 2^(-1022) }

```

```
19811: MaxLog = 709.7827128933840;
19812: MinLog = -708.3964185322641;
19813: MaxFac = 170;
19814: MaxGam = 171.624376956302;
19815: MaxLgm = 2.556348E+305;
19816: SingleCompareDelta = 1.0E-34;
19817: DoubleCompareDelta = 1.0E-280;
19818: {$IFDEF CLR}
19819: ExtendedCompareDelta = DoubleCompareDelta;
19820: {$ELSE}
19821: ExtendedCompareDelta = 1.0E-4400;
19822: {$ENDIF}
19823: Bytes1KB = 1024;
19824: Bytes1MB = 1024 * Bytes1KB;
19825: Bytes1GB = 1024 * Bytes1MB;
19826: Bytes64KB = 64 * Bytes1KB;
19827: Bytes64MB = 64 * Bytes1MB;
19828: Bytes2GB = 2 * LongWord(Bytes1GB);
19829: clBlack32', $FF000000 );
19830: clDimGray32', $FF3F3F3F );
19831: clGray32', $FF7F7F7F );
19832: clLightGray32', $FFBFBFBF );
19833: clWhite32', $FFFFFFFF );
19834: clMaroon32', $FF7F0000 );
19835: clGreen32', $FF007F00 );
19836: clOlive32', $FF7F7F00 );
19837: clNavy32', $FF00007F );
19838: clPurple32', $FF7F007F );
19839: clTeal32', $FF007F7F );
19840: clRed32', $FFFF0000 );
19841: clLime32', $FF00FF00 );
19842: clYellow32', $FFFFFF00 );
19843: clBlue32', $FF0000FF );
19844: clFuchsia32', $FFFF00FF );
19845: clAqua32', $FF00FFFF );
19846: clAliceBlue32', $FFF0F8FF );
19847: clAntiqueWhite32', $FFFAEBD7 );
19848: clAquamarine32', $FF7FFFD4 );
19849: clAzure32', $FFF0FFFF );
19850: clBeige32', $FFF5F5DC );
19851: clBisque32', $FFFFE4C4 );
19852: clBlancheDalmond32', $FFFFEBED );
19853: clBlueViolet32', $FF8A2BE2 );
19854: clBrown32', $FFA52A2A );
19855: clBurlyWood32', $FFDEB887 );
19856: clCadetblue32', $FF5F9EA0 );
19857: clChartReuse32', $FF7FFF00 );
19858: clChocolate32', $FFD2691E );
19859: clCoral32', $FFFF7F50 );
19860: clCornFlowerBlue32', $FF6495ED );
19861: clCornSilk32', $FFFFF8DC );
19862: clCrimson32', $FFDC143C );
19863: clDarkBlue32', $FF00008B );
19864: clDarkCyan32', $FF008B8B );
19865: clDarkGoldenRod32', $FFB8860B );
19866: clDarkGray32', $FFA9A9A9 );
19867: clDarkGreen32', $FF006400 );
19868: clDarkGrey32', $FFA9A9A9 );
19869: clDarkKhaki32', $FFBDB76B );
19870: clDarkMagenta32', $FF8B008B );
19871: clDarkOliveGreen32', $FF556B2F );
19872: clDarkOrange32', $FFF7F8C00 );
19873: clDarkOrchid32', $FF9932CC );
19874: clDarkRed32', $FF8B0000 );
19875: clDarkSalmon32', $FFE9967A );
19876: clDarkSeaGreen32', $FF8FBBC8F );
19877: clDarkSlateBlue32', $FF483D8B );
19878: clDarkSlateGray32', $FF2F4F4F );
19879: clDarkSlateGrey32', $FF2F4F4F );
19880: clDarkTurquoise32', $FF00CED1 );
19881: clDarkViolet32', $FF9400D3 );
19882: clDeepPink32', $FFFF1493 );
19883: clDeepSkyBlue32', $FF00BFFF );
19884: clDodgerBlue32', $FF1E90FF );
19885: clFireBrick32', $FFB22222 );
19886: clFloralWhite32', $FFFFFA00 );
19887: clGainsboro32', $FFDCDCDC );
19888: clGhostWhite32', $FFF8F8FF );
19889: clGold32', $FFFFD700 );
19890: clGoldenRod32', $FFDA520 );
19891: clGreenYellow32', $FFADFF2F );
19892: clGrey32', $FF808080 );
19893: clHoneyDew32', $FFFF69B4 );
19894: clHotPink32', $FFFF69B4 );
19895: clIndianRed32', $FFCD5C5C );
19896: clIndigo32', $FF4B0082 );
19897: clIvory32', $FFFFFFF0 );
19898: clKhaki32', $FFFOE68C );
19899: clLavender32', $FFE6E6FA );
```

```

19900:    clLavenderBlush32', $FFFFF0F5 ));  

19901:    clLawnGreen32', $FF7CFC00 ));  

19902:    clLemonChiffon32', $FFFFFFACD ));  

19903:    clLightBlue32', $FFFADD8E6 ));  

19904:    clLightCoral32', $FFF08080 ));  

19905:    clLightCyan32', $FFE0FFF ));  

19906:    clLightGoldenRodYellow32', $FFFFAFAD2 ));  

19907:    clLightGreen32', $FF90EE90 ));  

19908:    clLightGrey32', $FFD3D3D3 ));  

19909:    clLightPink32', $FFFFB6C1 ));  

19910:    clLightSalmon32', $FFFFA07A ));  

19911:    clLightSeagreen32', $FF20B2AA ));  

19912:    clLightSkyblue32', $FF87CEFA ));  

19913:    clLightSlategray32', $FF778899 ));  

19914:    clLightSlategrey32', $FF778899 ));  

19915:    clLightSteelblue32', $FFFB0C4DE ));  

19916:    clLightYellow32', $FFFFFE0 ));  

19917:    clLtGray32', $FFC0C0C0 ));  

19918:    clMedGray32', $FFA0A0A4 ));  

19919:    clDkGray32', $FF808080 ));  

19920:    clMoneyGreen32', $FFC0DCC0 ));  

19921:    clLegacySkyBlue32', $FFA6CAF0 ));  

19922:    clCream32', $FFFFFBF0 ));  

19923:    clLimeGreen32', $FF32CD32 ));  

19924:    clLinen32', $FFFAF0E6 ));  

19925:    clMediumAquamarine32', $FF66CDAA ));  

19926:    clMediumBlue32', $FF0000CD ));  

19927:    clMediumOrchid32', $FFBA55D3 ));  

19928:    clMediumPurple32', $FF9370DB ));  

19929:    clMediumSeaGreen32', $FF3CB371 ));  

19930:    clMediumSlateBlue32', $FF7B68EE ));  

19931:    clMediumSpringGreen32', $FF00FA9A ));  

19932:    clMediumTurquoise32', $FF48D1CC ));  

19933:    clMediumVioletRed32', $FFC71585 ));  

19934:    clMidnightBlue32', $FF191970 ));  

19935:    clMintCream32', $FFF5FFA ));  

19936:    clMistyRose32', $FFFFE4E1 ));  

19937:    clMoccasin32', $FFFFE4B5 ));  

19938:    clNavajoWhite32', $FFFFDEAD ));  

19939:    clOldLace32', $FFFD5E6 ));  

19940:    clOliveDrab32', $FF688E23 ));  

19941:    clOrange32', $FFFA500 ));  

19942:    clOrangeRed32', $FFFF4500 ));  

19943:    clOrchid32', $FFDA70D6 ));  

19944:    clPaleGoldenRod32', $FFEEE8AA ));  

19945:    clPaleGreen32', $FF98FB98 ));  

19946:    clPaleTurquoise32', $FFFAFEEE ));  

19947:    clPaleVioletred32', $FFDB7093 ));  

19948:    clPapayaWhip32', $FFFFEFD5 ));  

19949:    clPeachPuff32', $FFFFDAB9 ));  

19950:    clPeru32', $FFCD853F ));  

19951:    clPlum32', $FFDDA0DD ));  

19952:    clPowderBlue32', $FFB0E0E6 ));  

19953:    clRosyBrown32', $FFBC8F8F ));  

19954:    clRoyalBlue32', $FF4169E1 ));  

19955:    clSaddleBrown32', $FF8B4513 ));  

19956:    clSalmon32', $FFFA8072 ));  

19957:    clSandyBrown32', $FFF4A460 ));  

19958:    clSeaGreen32', $FF2E8B57 ));  

19959:    clSeaShell32', $FFFFF5EE ));  

19960:    clSienna32', $FFA0522D ));  

19961:    clSilver32', $FFC0C0C0 ));  

19962:    clSkyblue32', $FF87CEEB ));  

19963:    clSlateBlue32', $FF6A5ACD ));  

19964:    clSlateGray32', $FF708090 ));  

19965:    clSlateGrey32', $FF708090 ));  

19966:    clSnow32' $FFFFFAFA ));  

19967:    clSpringgreen32', $FF00FF7F ));  

19968:    clSteelblue32', $FF4682B4 ));  

19969:    clTan32', $FFD2B48C ));  

19970:    clThistle32', $FFD8BF08 ));  

19971:    clTomato32', $FFFF6347 ));  

19972:    clTurquoise32', $FF40E0D0 ));  

19973:    clViolet32', $FFEE82EE ));  

19974:    clWheat32', $FFF5DEB3 ));  

19975:    clWhitesmoke32', $FFF5F5F5 ));  

19976:    clYellowgreen32', $FF9ACD32 ));  

19977:    clTrWhite32', $7FFFFFF ));  

19978:    clTrBlack32', $7F000000 ));  

19979:    clTrRed32', $7FFF0000 ));  

19980:    clTrGreen32', $7F00FF00 ));  

19981:    clTrBlue32', $7F0000FF ));  

19982: // Fixed point math constants  

19983: FixedOne = $10000; FixedHalf = $7FFF;  

19984: FixedPI = Round(PI * FixedOne);  

19985: FixedToFloat = 1/FixedOne;  

19986:  

19987: Special Types  

19988: ****

```

```

19989: type Complex = record
19990:   X, Y : Float;
19991: end;
19992: type TVector     = array of Float;
19993: TIntVector    = array of Integer;
19994: TCompVector   = array of Complex;
19995: TBoolVector   = array of Boolean;
19996: TStringVector = array of String;
19997: TMatrix        = array of TVector;
19998: TIntMatrix    = array of TIntVector;
19999: TCompMatrix   = array of TCompVector;
20000: TBoolMatrix   = array of TBoolVector;
20001: TStringMatrix = array of TStringVector;
20002: TByteArray    = array[0..32767] of byte; !
20003: THexArray     = array [0..15] of Char; // = '0123456789ABCDEF';
20004: TBitmapStyle  = (bsNormal, bsCentered, bsStretched);
20005: T2StringArray = array of array of string;
20006: T2IntegerArray= array of array of integer;
20007: AddTypes('INT_PTR', 'Integer');
20008: AddTypes('LONG_PTR', 'Integer');
20009: AddTypes('UINT_PTR', 'Cardinal');
20010: AddTypeS('ULONG_PTR', 'Cardinal');
20011: AddTypeS('DWORD_PTR', 'ULONG_PTR');
20012: TInt64DynArray', 'array of Integer;
20013: TCardinalDynArray', 'array of Cardinal;
20014: TWordDynArray', 'array of Word;
20015: TSmallIntDynArray', 'array of SmallInt;
20016: TByteDynArray', 'array of Byte;
20017: TShortIntDynArray', 'array of ShortInt;
20018: TInt64DynArray', 'array of Int64;
20019: TLongWordDynArray', 'array of LongWord;
20020: TSingleDynArray', 'array of Single;
20021: TDoubleDynArray', 'array of Double;
20022: TBooleanDynArray', 'array of Boolean;
20023: TStringDynArray', 'array of string;
20024: TWideStringDynArray', 'array of WideString;
20025: TDynByteArray = array of Byte;
20026: TDynShortintArray = array of Shortint;
20027: TDynSmallintArray = array of Smallint;
20028: TDynWordArray = array of Word;
20029: TDynIntegerArray = array of Integer;
20030: TDynLongintArray = array of Longint;
20031: TDynCardinalArray = array of Cardinal;
20032: TDynInt64Array = array of Int64;
20033: TDynExtendedArray = array of Extended;
20034: TDynDoubleArray = array of Double;
20035: TDynSingleArray = array of Single;
20036: TDynFloatArray = array of Float;
20037: TDynPointerArray = array of Pointer;
20038: TDynStringArray = array of string;
20039: TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
20040:           ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
20041: TSynSearchOptions = set of TSynSearchOption;
20042:
20043:
20044: /* Project : Base Include RunTime Lib for maxbox *Name: pas_includebox.inc
20045: -----
20046: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
20047: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
20048: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
20049: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
20050: function CheckStringSum(vstring: string): integer;
20051: function HexToInt(HexNum: string): LongInt;
20052: function IntToBin(Int: Integer): String;
20053: function BinToInt(Binary: String): Integer;
20054: function HexToBin(HexNum: string): string; external2;
20055: function BinToHex(Binary: String): string;
20056: function IntToFloat(i: Integer): double;
20057: function AddThousandSeparator(S: string; myChr: Char): string;
20058: function Max3(const X,Y,Z: Integer): Integer;
20059: procedure Swap(var X,Y: char); // faster without inline;
20060: procedure ReverseString(var S: String);
20061: function CharToHexStr(Value: Char): string;
20062: function CharToUniCode(Value: Char): string;
20063: function Hex2Dec(Value: Str002): Byte;
20064: function HexStrCodeToStr(Value: string): string;
20065: function HexToStr(i: integer; value: string): string;
20066: function UniCodeToStr(Value: string): string;
20067: function CRC16(statement: string): string;
20068: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
20069: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
20070: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
20071: Procedure ExecuteCommand(executeFile, paramstring: string);
20072: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
20073: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
20074: procedure SearchAndOpenDoc(vfilenamepath: string);
20075: procedure ShowInterfaces(myFile: string);
20076: function Fact2(av: integer): extended;
20077: Function BoolToStr(B: Boolean): string;

```

```

20078: Function GCD(x, y : LongInt) : LongInt;
20079: function LCM(m,n: longint): longint;
20080: function GetASCII: string;
20081: function GetItemHeight(Font: TFont): Integer;
20082: function myPlaySound(s: pchar; flag, syncflag: integer): boolean;
20083: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
20084: function getHINSTANCE: longword;
20085: function getHMODULE: longword;
20086: function GetASCII: string;
20087: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
20088: function WordIsOk(const AWord: string; var VW: Word): boolean;
20089: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
20090: function LongIsOk(const Along: string; var VC: Cardinal): boolean;
20091: function SafeStr(const s: string): string;
20092: function ExtractUrlPath(const FileName: string): string;
20093: function ExtractUrlName(const FileName: string): string;
20094: function IsInternet: boolean;
20095: function RotateLeft1Bit_u32( Value: uint32): uint32;
20096: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double:NData:Int;var LF:TStLinEst; ErrorStats: Bool);
20097: procedure getEnvironmentInfo;
20098: procedure AntiFreeze;
20099: function GetCPUSpeed: Double;
20100: function IsVirtualPcGuest : Boolean;
20101: function IsVmWareGuest : Boolean;
20102: procedure StartSerialDialog;
20103: function IsWoW64: boolean;
20104: function IsWow64String(var s: string): Boolean;
20105: procedure StartThreadDemo;
20106: Function RGB(R,G,B: Byte): TColor;
20107: Function Sendln(amess: string): boolean;
20108: Procedure maxbox;
20109: Function AspectRatio(aWidth, aHeight: Integer): String;
20110: function wget(aURL, afile: string): boolean;
20111: procedure PrintList(Value: TStringList);
20112: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
20113: procedure getEnvironmentInfo;
20114: procedure AntiFreeze;
20115: function getBitmap(apath: string): TBitmap;
20116: procedure ShowMessageBig(const aText : string);
20117: function YesNoDialog(const ACaption, AMsg: string): boolean;
20118: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
20119: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
20120: //function myStrToBytes(const Value: String): TBytes;
20121: //function myBytesToStr(const Value: TBytes): String;
20122: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20123: function getBitmap(apath: string): TBitmap;
20124: procedure ShowMessageBig(const aText : string);
20125: Function StrToBytes(const Value: String): TBytes;
20126: Function BytesToStr(const Value: TBytes): String;
20127: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20128: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retries:Int;var HostName:String):Bool;
20129: function FindInPaths(const fileName, paths : String) : String;
20130: procedure initHexArray(var hexn: THexArray);
20131: function josephusG(n,k: integer; var graphout: string): integer;
20132: function isPowerof2(num: int64): boolean;
20133: function powerOf2(exponent: integer): int64;
20134: function getBigPI: string;
20135: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
20136: function GetASCIILine: string;
20137: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
20138: pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
20139: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
20140: procedure AddComplexSoundObjectToList(newf,newp,newa,neww:integer; freqlist: TStrings);
20141: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
20142: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
20143: function isKeypressed: boolean;
20144: function Keypress: boolean;
20145: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
20146: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
20147: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
20148: function GetOSName: string;
20149: function GetOSVersion: string;
20150: function GetOSNumber: string;
20151: function getEnvironmentString: string;
20152: procedure StrReplace(var Str: String; Old, New: String);
20153: procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
20154: function getTeamViewerID: string;
20155: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFFileCallbackProcedure);
20156: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
20157: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
20158: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
20159: function StartSocketService: Boolean;
20160: procedure StartSocketServiceForm;
20161: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
20162: function GetFileList1(apath: string): TStringlist;
20163: procedure LetFileList(FileList: TStringlist; apath: string);
20164: procedure StartWeb(aurl: string);

```

```

20165: function GetTodayFiles(startdir, amask: string): TStringlist;
20166: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
20167: function JavahashCode(val: string): Integer;
20168: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
20169: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
20170: Procedure HideWindowForSeconds(secs: integer); //3 seconds
20171: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); //3 seconds
20172: Procedure ConvertToGray(Cnv: TCanvas);
20173: function GetFileDate(aFile:string; aWithTime:Boolean):string;
20174: procedure ShowMemory;
20175: function ShowMemory2: string;
20176: function getHostIP: string;
20177: procedure ShowBitmap(bmap: TBitmap);
20178: function GetOsVersionInfo: TOSversionInfo; //thx to wischnewski
20179: function CreateDBGridForm(dblist: TStringList): TListBox;
20180: function isService: boolean;
20181: function isApplication: boolean;
20182: function isTerminalSession: boolean;
20183:
20184:
20185: // News of 3.9.8 up
20186: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20187: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20188: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20189: JvChart - TJvChart Component - 2009 Public
20190: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
20191: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
20192: TAdoQuery.SQL.Add() fixed, ShlwAPI extensions, Indy HTTPHeader Extensions
20193: DMath DLL included incl. Demos
20194: Interface Navigator menu/View/Intf Navigator
20195: Unit Explorer menu/Debug/Units Explorer
20196: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
20197: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
20198: Script History to 9 Files WebServer light /Options/Addons/WebServer
20199: Full Text Finder, JVSsimLogic Simulator Package
20200: Halt-Stop Program in Menu, WebServer2, Stop Event ,
20201: Conversion Routines, Prebuild Forms, CodeSearch
20202: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20203: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20204: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20205: JvChart - TJvChart Component - 2009 Public, mxGames, JvgXMLLoader, TJvPaintFX
20206: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
20207: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
20208: IDE Reflection API, Session Service Shell S3
20209: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
20210: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
20211: arduino map() function, PRandom Generator
20212: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
20213: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
20214: REST Test Lib, Multilang Component, Forth Interpreter
20215: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
20216: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
20217: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20218: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20219: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
20220: QRCode Service, add more CFunctions like CDateTime of Synapse
20221: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
20222: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
20223: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
20224: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
20225: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
20226: BOLD Package, Indy Package5, maTRIX, MATHEMAX
20227: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
20228: emax layers: system-package-component-unit-class-function-block
20229: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms
20230: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
20231: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
20232: OpenGL Game Demo: ..Options/Add Ons/Reversi
20233: IUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
20234: add 31 units, mx4 Introduction Paper, more Socket&Streams, ShortString Routines
20235: 7% performance gain (hot spot profiling)
20236: PEP -Pascal Education Program, GSM Module, CGI, PHP Runner
20237: add 42 + 22 (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
20238: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
20239: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
20240:
20241: add routines in 3.9.7.5
20242: 097: procedure RIRegister_BarCodeScanner_Routines(S: TPSEExec);
20243: 996: procedure RIRegister_DBCtrls_Routines(S: TPSEExec);
20244: 069: procedure RIRegister_IdStrings_Routines(S: TPSEExec);
20245: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSEExec);
20246: 215: procedure RIRegister_PNGLoader_Routines(S: TPSEExec);
20247: 374: procedure RIRegister_SerDlgls_Routines(S: TPSEExec);
20248: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSEExec);
20249:
20250: ////////////////////////////// TestUnits //////////////////////////////
20251: SelftestPEM;
20252: SelfTestCFundamentUtils;
20253: SelfTestCFileUtils;

```

```

20254: SelfTestCDatetime;
20255: SelfTestCTimer;
20256: SelfTestCRandom;
20257: Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter'
20258:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath'
20259:
20260: // Note: There's no need for installing a client certificate in the
20261: // webbrowser. The server asks the webbrowser to send a certificate but
20262: // if nothing is installed the software will work because the server
20263: // doesn't check to see if a client certificate was supplied. If you want you can install:
20264: // file: c_cacert.p12 password: c_cakey
20265:
20266: TGraphicControl = class(TControl)
20267: private
20268:   FCanvas: TCanvas;
20269:   procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20270: protected
20271:   procedure Paint; virtual;
20272:   property Canvas: TCanvas read FCanvas;
20273: public
20274:   constructor Create(AOwner: TComponent); override;
20275:   destructor Destroy; override;
20276: end;
20277:
20278: TCustomControl = class(TWinControl)
20279: private
20280:   FCanvas: TCanvas;
20281:   procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20282: protected
20283:   procedure Paint; virtual;
20284:   procedure PaintWindow(DC: HDC); override;
20285:   property Canvas: TCanvas read FCanvas;
20286: public
20287:   constructor Create(AOwner: TComponent); override;
20288:   destructor Destroy; override;
20289: end;
20290: RegisterPublishedProperties;
20291: ('ONCHANGE', 'TNotifyEvent', iptrw);
20292: ('ONCLICK', 'TNotifyEvent', iptrw);
20293: ('ONDBLCLICK', 'TNotifyEvent', iptrw);
20294: ('ONENTER', 'TNotifyEvent', iptrw);
20295: ('ONEXIT', 'TNotifyEvent', iptrw);
20296: ('ONKEYDOWN', 'TKeyEvent', iptrw);
20297: ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
20298: ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
20299: ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
20300: ('ONMOUSEUP', 'TMouseEvent', iptrw);
20301: //*****
20302: // To stop the while loop, click on Options>Show Include (boolean switch) !
20303: Control a loop in a script with a form event:
20304: IncludeON; //control the while loop
20305: while maxform1.ShowInclude1.checked do begin //menu event Options>Show Include
20306: repeat {for it:= 1 to n do} until is keypressed //keypress in output window below (memo2)
20307:
20308: //-----
20309: //*****mX4 ini-file Configuration*****
20310: //-----
20311: using config file maxboxdef.ini      menu/Help/Config File
20312:
20313: //*** Definitions for maxbox mX3 ***
20314: [FORM]
20315: LAST_FILE=E:\maxbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
20316: FONTSIZE=14
20317: EXTENSION=txt
20318: SCREENX=1386
20319: SCREENY=1077
20320: MEMHEIGHT=350
20321: PRINTFONT=Courier New //GUI Settings
20322: LINENUMBERS=Y //line numbers at gutter in editor at left side
20323: EXCEPTIONLOG=Y //store excepts + success in 2 log files see below! -menu Debug>Show Last Exceptions
20324: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
20325: BOOTSCRIPT=Y //enabling load a boot script
20326: MEMORYREPORT=Y //shows memory report on closing maxbox
20327: MACRO=Y //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
20328: NAVIGATOR=N //shows function list at the right side of editor
20329: NAVWIDTH=350 //width of the right side interface list <CTRL L> >=200
20330: AUTOBOOKMARK=Y //sets on all functions a bookmark to jump
20331: [WEB]
20332: IPPORT=8080 //for internal webserver - menu /Options/Add Ons/WebServer
20333: IPHOST=192.168.1.53
20334: ROOTCERT=filepathY
20335: SCERT=filepathY
20336: RSAKEY=filepathY
20337: VERSIONCHECK=Y
20338:
20339: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
20340: Also possible to set report memory in script to override ini setting
20341: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
20342:

```

```

20343: After Change the ini file you can reload the file with ..\Help\Config Update
20344:
20345: -----
20346: //*****mX4 maildef.ini ini-file Configuration*****
20347: -----
20348: **** Definitions for maxMail ***
20349: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
20350: [MAXMAIL]
20351: HOST=mailto.softwareschule.ch
20352: USER=mailusername
20353: PASS=password
20354: PORT=110
20355: SSL=Y
20356: BODY=Y
20357: LAST=5
20358:
20359: ADO Connection String:
20360: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
20361: \452_dbtreeview2access
20362: program TestDbTreeViewMainForm2_ACCESS;
20363:   ConnectionString='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
20364:           +Exepath+'\examples\detail.mdb;Persist Security Info=False';
20365:
20366: OpenSSL Lib: unit ssl_openssl_lib;
20367: {$IFDEF CIL}
20368: const
20369: {$IFDEF LINUX}
20370: DLLSSName = 'libssl.so';
20371: DLLUtilName = 'libcrypto.so';
20372: {$ELSE}
20373: DLLSSName = 'ssleay32.dll';
20374: DLLUtilName = 'libeay32.dll';
20375: {$ENDIF}
20376: {$ELSE}
20377: var
20378: {$IFNDEF MSWINDOWS}
20379: {$IFDEF DARWIN}
20380: DLLSSName: string = 'libssl.dylib';
20381: DLLUtilName: string = 'libcrypto.dylib';
20382: {$ELSE}
20383: DLLSSName: string = 'libssl.so';
20384: DLLUtilName: string = 'libcrypto.so';
20385: {$ENDIF}
20386: {$ELSE}
20387: DLLSSName: string = 'ssleay32.dll';
20388: DLLSSName2: string = 'libssl32.dll';
20389: DLLUtilName: string = 'libeay32.dll';
20390: {$ENDIF}
20391: {$ENDIF}
20392:
20393:
20394: -----
20395: //*****mX4 Macro Tags *****
20396: -----
20397:
20398: asm #name #hostmAPSN2APSN21l1e, #head,max: APSN21: 04.01.2014 19:05:50
E:\maxbox\maxbox3\docs\maxbox_extract_funcList399.txt end
20399:
20400: //Tag Macros
20401:
20402: asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
20403:
20404: //Tag Macros
20405: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
20406: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
20407: 10190: SearchAndCopy(memo1.lines, '#host', getComputerNameWin, 11);
20408: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
20409: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
20410: 10199: SearchAndCopy(memo1.lines, '#fils', fname + '\'+SHA1(Act_Filename), 11);
20411: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
20412: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
20413: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
20414: [getUserNameWin, getComputerNameWin, datetimetoStr(now),
20415: 10196: SearchAndCopy(memo1.lines, '#head', Format('%s: %s: %s',
20416: 10197: [getUserNameWin, getComputerNameWin, datetimetoStr(now), Act_Filename]], 11);
20417: [getUserNameWin, getComputerNameWin, datetimetoStr(now), Act_Filename]], 11);
20418: 10198: SearchAndCopy(memo1.lines, '#tech', Format('perf: %s threads: %d %s %s',
20419: [perftime, numprocesssthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]], 11);
20420: 10298: SearchAndCopy(memo1.lines, '#net', Format('DNS: %s; local IPs: %s; local IP: %s',
20421: [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]], 10);
20422:
20423: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
20424:
20425: //Replace Macros
20426: SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
20427: SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
20428: SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
20429: SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPATH, 9);
20430: SearchAndCopy(memo1.lines, '<FILE>', fname, 6);

```

```

20431: SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
20432:
20433: SearchAndCopy(memo1.lines, '#tech'perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
20434: [perftime,numprocesssthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]], 11);
20435: //#tech!84perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
20436: SearchAndCopy(memo1.lines, 'maxbox_extract_funcList399.txt'
20437:
20438: //-----
20439: //*****mX4 ToDo List Tags ..../Help/ToDo List*****
20440: //-----
20441:
20442: while I < sl.Count do begin
20443:   if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9#]*:*)' ) then
20444:     if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
20445:       BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
20446:     else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
20447:       BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
20448:     else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
20449:       BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
20450:     else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
20451:       BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
20452:     else if MatchesMask(sl[I], '*/?TODO*:*)' ) then
20453:       BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
20454:     else if MatchesMask(sl[I], '*/?DONE*:*)' ) then
20455:       BreakupToDo(Filename, sl, I, 'DONE', False, False) // custom DONE
20456:     Inc(I);
20457:   end;
20458:
20459: //-----
20460: //*****mX4 Public Tools API *****
20461: //-----
20462: file : unit uPSI_fMain.pas;                                {$OTAP} Open Tools API Catalog
20463: // Those functions concern the editor and preprocessor, all of the IDE
20464: Example: Call it with maxform1.InfoClick(self)
20465: Note: Call all Methods with maxForm1., e.g.:
20466:           maxForm1.ShellStyle1Click(self);
20467:
20468: procedure SIRegister_fMain(CL: TPSPascalCompiler);
20469: begin
20470:   Const ('BYTECODE', 'String 'bytecode.txt'
20471:   Const ('PSTEXT', 'String PS Scriptfiles (*.txt)|*.TXT)'
20472:   Const ('PSMODEL', 'String PS Modelfiles (*.uc)|*.UC'
20473:   Const ('PSPASCAL', 'String PS Pascalfiles (*.pas)|*.PAS'
20474:   Const ('PSINC', 'String PS Includes (*.inc)|*.INC'
20475:   Const ('DEFFILENAME', 'String 'firstdemo.txt'
20476:   Const ('DEFINIFILE', 'String 'maxboxdef.ini'
20477:   Const ('EXCEPTLOGFILE', 'String 'maxboxerrorlog.txt'
20478:   Const ('ALLFUNCTIONSLIST', 'String 'ups1_allfunctionslist.txt'
20479:   Const ('ALLFUNCTIONSLISTPDF', 'String 'maxbox_functions_all.pdf'
20480:   Const ('ALLOBJECTSLIST', 'String 'docs\VCL.pdf'
20481:   Const ('ALLRESOURCELIST', 'String 'docs\psi1_allresourcelist.txt'
20482:   Const ('ALLUNITLIST', 'String 'docs\maxbox3_9.xml');
20483:   Const ('INCLUDEBOX', 'String 'pas_includebox.inc'
20484:   Const ('BOOTSCRIPT', 'String 'maxbootscript.txt'
20485:   Const ('MBVERSION', 'String '3.9.9.101
20486:   Const ('VERSION', 'String '3.9.9.101
20487:   Const ('MBVER', 'String '399
20488:   Const ('MBVER1', 'Integer'(399);
20489:   Const ('MBVERIALL', 'Integer'(399101);
20490:   Const ('EXENAME', 'String 'maxBox3.exe
20491:   Const ('MXSITE', 'String 'http://www.softwareschule.ch/maxbox.htm
20492:   Const ('MXVERSIONFILE', 'String 'http://www.softwareschule.ch/maxvfile.txt
20493:   Const ('MXVERSIONFILE2', 'String 'http://www.softwareschule.ch/maxvfile2.txt
20494:   Const ('MXINTERNETCHECK', 'String 'www.ask.com
20495:   Const ('MXMAIL', 'String 'max@kleiner.com
20496:   Const ('TAB', 'Char #$09);
20497:   Const ('CODECOMPLETION', 'String 'bds_delphi.dci
20498:   SIRegister_TMaxForm1(CL);
20499: end;
20500:
20501: with FindClass('TForm'),'TMaxForm1') do begin
20502:   memo2, 'TMemo', iptrw;
20503:   memo1, 'TSynMemo', iptrw;
20504:   CB1SList1, 'TComboBox', iptrw;
20505:   mxNavigator1, 'TComboBox', iptrw);
20506:   IPHost1, 'string', iptrw);
20507:   IPPort1, 'integer', iptrw);
20508:   COMPort1, 'integer', iptrw);      //3.9.6.4
20509:   Splitter1, 'TSplitter', iptrw);
20510:   PSScript1, 'TPSScript', iptrw);
20511:   PS3DllPlugin1, 'TPSDllPlugin', iptrw);
20512:   MainMenul1, 'TMainMenu', iptrw);
20513:   Program1, 'TMenuItem', iptrw);
20514:   Compile1, 'TMenuItem', iptrw);
20515:   Files1, 'TMenuItem', iptrw);
20516:   open1, 'TMenuItem', iptrw);
20517:   Save2, 'TMenuItem', iptrw);
20518:   Options1, 'TMenuItem', iptrw);
20519:   Savebefore1, 'TMenuItem', iptrw);

```

```
20520: Largefont1', 'TMenuItem', iptrw);
20521: sBytecode1', 'TMenuItem', iptrw);
20522: Saveas3', 'TMenuItem', iptrw);
20523: Clear1', 'TMenuItem', iptrw);
20524: Slinenumbers1', 'TMenuItem', iptrw);
20525: About1', 'TMenuItem', iptrw);
20526: Search1', 'TMenuItem', iptrw);
20527: SynPasSyn1', 'TSynPasSyn', iptrw);
20528: memo1', 'TSynMemo', iptrw);
20529: SynEditSearch1', 'TSynEditSearch', iptrw);
20530: WordWrap1', 'TMenuItem', iptrw);
20531: XPMManifest1', 'TXPManifest', iptrw);
20532: SearchNext1', 'TMenuItem', iptrw);
20533: Replace1', 'TMenuItem', iptrw);
20534: PSImport_Controls1', 'TPSImport_Controls', iptrw);
20535: PSImport_Classes1', 'TPSImport_Classes', iptrw);
20536: ShowInclude1', 'TMenuItem', iptrw);
20537: SynEditPrint1', 'TSynEditPrint', iptrw);
20538: Printout1', 'TMenuItem', iptrw);
20539: mnPrintColors1', 'TMenuItem', iptrw);
20540: dlgFilePrint', 'TPrintDialog', iptrw);
20541: dlgPrintFont1', 'TFontDialog', iptrw);
20542: mnuPrintFont1', 'TMenuItem', iptrw);
20543: Include1', 'TMenuItem', iptrw);
20544: CodeCompletionList1', 'TMenuItem', iptrw);
20545: IncludeList1', 'TMenuItem', iptrw);
20546: ImageList1', 'TImageList', iptrw);
20547: ImageList2', 'TImageList', iptrw);
20548: CoolBar1', 'TCoolBar', iptrw);
20549: ToolBar1', 'TToolBar', iptrw);
20550: btnLoad', 'TToolButton', iptrw);
20551: ToolButton2', 'TToolButton', iptrw);
20552: btnFind', 'TToolButton', iptrw);
20553: btnCompile', 'TToolButton', iptrw);
20554: btnTrans', 'TToolButton', iptrw);
20555: btnUseCase', 'TToolButton', iptrw); //3.8
20556: toolbtnTutorial', 'TToolButton', iptrw);
20557: btn6res', 'TToolButton', iptrw);
20558: ToolButton5', 'TToolButton', iptrw);
20559: ToolButton1', 'TToolButton', iptrw);
20560: ToolButton3', 'TToolButton', iptrw);
20561: statusBar1', 'TStatusBar', iptrw);
20562: SaveOutput1', 'TMenuItem', iptrw);
20563: ExportClipboard1', 'TMenuItem', iptrw);
20564: Close1', 'TMenuItem', iptrw);
20565: Manual1', 'TMenuItem', iptrw);
20566: About2', 'TMenuItem', iptrw);
20567: loadLastfile1', 'TMenuItem', iptrw);
20568: imgLogo', 'TImage', iptrw);
20569: cedebbug', 'TPSScriptDebugger', iptrw);
20570: debugPopupMenu1', 'TPopupMenu', iptrw);
20571: BreakPointMenu', 'TMenuItem', iptrw);
20572: Decompile1', 'TMenuItem', iptrw);
20573: StepInto1', 'TMenuItem', iptrw);
20574: StepOut1', 'TMenuItem', iptrw);
20575: Reset1', 'TMenuItem', iptrw);
20576: DebugRun1', 'TMenuItem', iptrw);
20577: PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
20578: PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
20579: PSImport_Forms1', 'TPSImport_Forms', iptrw);
20580: PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
20581: tutorial4', 'TMenuItem', iptrw);
20582: ExporttoClipboard1', 'TMenuItem', iptrw);
20583: ImportfromClipboard1', 'TMenuItem', iptrw);
20584: N4', 'TMenuItem', iptrw);
20585: N5', 'TMenuItem', iptrw);
20586: N6', 'TMenuItem', iptrw);
20587: ImportfromClipboard2', 'TMenuItem', iptrw);
20588: tutorial1', 'TMenuItem', iptrw);
20589: N7', 'TMenuItem', iptrw);
20590: ShowSpecChars1', 'TMenuItem', iptrw);
20591: OpenDirectory1', 'TMenuItem', iptrw);
20592: procMess', 'TMenuItem', iptrw);
20593: btnUseCase', 'TToolButton', iptrw);
20594: ToolButton7', 'TToolButton', iptrw);
20595: EditFont1', 'TMenuItem', iptrw);
20596: UseCase1', 'TMenuItem', iptrw);
20597: tutorial21', 'TMenuItem', iptrw);
20598: OpenUseCase1', 'TMenuItem', iptrw);
20599: PSImport_DB1', 'TPSImport_DB', iptrw);
20600: tutorial31', 'TMenuItem', iptrw);
20601: SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
20602: HTMLSyntax1', 'TMenuItem', iptrw);
20603: ShowInterfaces1', 'TMenuItem', iptrw);
20604: Tutorials5', 'TMenuItem', iptrw);
20605: AllFunctionsList1', 'TMenuItem', iptrw);
20606: ShowLastException1', 'TMenuItem', iptrw);
20607: PlayMP31', 'TMenuItem', iptrw);
20608: SynTeXSyn1', 'TSynTeXSyn', iptrw);
```

```
20609: texSyntax1', 'TMenuItem', iptrw);
20610: N8', 'TMenuItem', iptrw);
20611: GetEMails1', 'TMenuItem', iptrw);
20612: SynCppSyn1', 'TSynCppSyn', iptrw);
20613: CSyntax1', 'TMenuItem', iptrw);
20614: Tutorial6', 'TMenuItem', iptrw);
20615: New1', 'TMenuItem', iptrw);
20616: AllObjectsList1', 'TMenuItem', iptrw);
20617: LoadBytecode1', 'TMenuItem', iptrw);
20618: CipherFile1', 'TMenuItem', iptrw);
20619: N9', 'TMenuItem', iptrw);
20620: N10', 'TMenuItem', iptrw);
20621: Tutorial11', 'TMenuItem', iptrw);
20622: Tutorial71', 'TMenuItem', iptrw);
20623: UpdateService1', 'TMenuItem', iptrw);
20624: PascalSchool1', 'TMenuItem', iptrw);
20625: Tutorial81', 'TMenuItem', iptrw);
20626: DelphiSite1', 'TMenuItem', iptrw);
20627: Output1', 'TMenuItem', iptrw);
20628: TerminalStyle1', 'TMenuItem', iptrw);
20629: ReadOnly1', 'TMenuItem', iptrw);
20630: ShellStyle1', 'TMenuItem', iptrw);
20631: BigScreen1', 'TMenuItem', iptrw);
20632: Tutorial91', 'TMenuItem', iptrw);
20633: SaveOutput2', 'TMenuItem', iptrw);
20634: N11', 'TMenuItem', iptrw);
20635: SaveScreenshot', 'TMenuItem', iptrw);
20636: Tutorial101', 'TMenuItem', iptrw);
20637: SQLSyntax1', 'TMenuItem', iptrw);
20638: SynSQLSyn1', 'TSynSQLSyn', iptrw);
20639: Console1', 'TMenuItem', iptrw);
20640: SynXMLSyn1', 'TSynXMLSyn', iptrw);
20641: XMLSyntax1', 'TMenuItem', iptrw);
20642: ComponentCount1', 'TMenuItem', iptrw);
20643: NewInstance1', 'TMenuItem', iptrw);
20644: toolbtnTutorial', 'TToolButton', iptrw);
20645: Memory1', 'TMenuItem', iptrw);
20646: SynJavaSyn1', 'TSynJavaSyn', iptrw);
20647: JavaSyntax1', 'TMenuItem', iptrw);
20648: SyntaxCheck1', 'TMenuItem', iptrw);
20649: Tutorial10Statistics1', 'TMenuItem', iptrw);
20650: ScriptExplorer1', 'TMenuItem', iptrw);
20651: FormOutput1', 'TMenuItem', iptrw);
20652: ArduinoDump1', 'TMenuItem', iptrw);
20653: AndroidDump1', 'TMenuItem', iptrw);
20654: GotoEnd1', 'TMenuItem', iptrw);
20655: AllResourceList1', 'TMenuItem', iptrw);
20656: ToolButton4', 'TToolButton', iptrw);
20657: btn6res', 'TToolButton', iptrw);
20658: Tutorial11Forms1', 'TMenuItem', iptrw);
20659: Tutorial12SQL1', 'TMenuItem', iptrw);
20660: ResourceExplore1', 'TMenuItem', iptrw);
20661: Info1', 'TMenuItem', iptrw);
20662: N12', 'TMenuItem', iptrw);
20663: CryptoBox1', 'TMenuItem', iptrw);
20664: Tutorial13Ciphering1', 'TMenuItem', iptrw);
20665: CipherFile2', 'TMenuItem', iptrw);
20666: N13', 'TMenuItem', iptrw);
20667: ModulesCount1', 'TMenuItem', iptrw);
20668: AddOns2', 'TMenuItem', iptrw);
20669: N4GewinntGame1', 'TMenuItem', iptrw);
20670: DocuforAddOns1', 'TMenuItem', iptrw);
20671: Tutorial14Async1', 'TMenuItem', iptrw);
20672: Lessons15Review1', 'TMenuItem', iptrw);
20673: SynPHPSyn1', 'TSynPHPSyn', iptrw);
20674: PHPSyntax1', 'TMenuItem', iptrw);
20675: Breakpoint1', 'TMenuItem', iptrw);
20676: SerialRS2321', 'TMenuItem', iptrw);
20677: N14', 'TMenuItem', iptrw);
20678: SynCSSyn1', 'TSynCSSyn', iptrw);
20679: CSyntax2', 'TMenuItem', iptrw);
20680: Calculator1', 'TMenuItem', iptrw);
20681: btnSerial', 'TToolButton', iptrw);
20682: ToolButton8', 'TToolButton', iptrw);
20683: Tutorial151', 'TMenuItem', iptrw);
20684: N15', 'TMenuItem', iptrw);
20685: N16', 'TMenuItem', iptrw);
20686: ControlBar1', 'TControlBar', iptrw);
20687: ToolBar2', 'TToolBar', iptrw);
20688: BtnOpen', 'TToolButton', iptrw);
20689: BtnSave', 'TToolButton', iptrw);
20690: BtnPrint', 'TToolButton', iptrw);
20691: BtnColors', 'TToolButton', iptrw);
20692: btnClassReport', 'TToolButton', iptrw);
20693: BtnRotateRight', 'TToolButton', iptrw);
20694: BtnFullSize', 'TToolButton', iptrw);
20695: BtnFitToWindowSize', 'TToolButton', iptrw);
20696: BtnZoomMinus', 'TToolButton', iptrw);
20697: BtnZoomPlus', 'TToolButton', iptrw);
```

```
20698: Panel1', 'TPanel', iptrw);
20699: LabelBrettgroesse', 'TLabel', iptrw);
20700: CB1SCList', 'TComboBox', iptrw);
20701: ImageListNormal', 'TImageList', iptrw);
20702: spbtnexplore', 'TSpeedButton', iptrw);
20703: spbtnexample', 'TSpeedButton', iptrw);
20704: spbsaveas', 'TSpeedButton', iptrw);
20705: imglogobox', 'TImage', iptrw);
20706: EnlargeFont1', 'TMenuItem', iptrw);
20707: EnlargeFont2', 'TMenuItem', iptrw);
20708: ShrinkFont1', 'TMenuItem', iptrw);
20709: ThreadDemo1', 'TMenuItem', iptrw);
20710: HEXEditor1', 'TMenuItem', iptrw);
20711: HEXView1', 'TMenuItem', iptrw);
20712: HEXInspect1', 'TMenuItem', iptrw);
20713: SynExporterHTML1', 'TSynExporterHTML', iptrw);
20714: ExporttoHTML1', 'TMenuItem', iptrw);
20715: ClassCount1', 'TMenuItem', iptrw);
20716: HTMLOutput1', 'TMenuItem', iptrw);
20717: HEXEditor2', 'TMenuItem', iptrw);
20718: Minesweeper1', 'TMenuItem', iptrw);
20719: N17', 'TMenuItem', iptrw);
20720: PicturePuzzle1', 'TMenuItem', iptrw);
20721: sbvchelp', 'TSpeedButton', iptrw);
20722: DependencyWalker1', 'TMenuItem', iptrw);
20723: WebScanner1', 'TMenuItem', iptrw);
20724: View1', 'TMenuItem', iptrw);
20725: mnToolbar1', 'TMenuItem', iptrw);
20726: mnStatusbar2', 'TMenuItem', iptrw);
20727: mnConsole2', 'TMenuItem', iptrw);
20728: mnCoolbar2', 'TMenuItem', iptrw);
20729: mnSplitter2', 'TMenuItem', iptrw);
20730: WebServer1', 'TMenuItem', iptrw);
20731: Tutorial17Server1', 'TMenuItem', iptrw);
20732: Tutorial18Arduino1', 'TMenuItem', iptrw);
20733: SynPerlSyn1', 'TSynPerlSyn', iptrw);
20734: PerlSyntax1', 'TMenuItem', iptrw);
20735: SynPythonSyn1', 'TSynPythonSyn', iptrw);
20736: PythonSyntax1', 'TMenuItem', iptrw);
20737: DMathLibrary1', 'TMenuItem', iptrw);
20738: IntfNavigator1', 'TMenuItem', iptrw);
20739: EnlargeFontConsole1', 'TMenuItem', iptrw);
20740: ShrinkFontConsole1', 'TMenuItem', iptrw);
20741: SetInterfaceList1', 'TMenuItem', iptrw);
20742: popintfList', 'TPopupMenu', iptrw);
20743: intfAdd1', 'TMenuItem', iptrw);
20744: intfDelete1', 'TMenuItem', iptrw);
20745: intfRefactor1', 'TMenuItem', iptrw);
20746: Defactor1', 'TMenuItem', iptrw);
20747: Tutorial19COMArduino1', 'TMenuItem', iptrw);
20748: Tutorial20Regex1', 'TMenuItem', iptrw);
20749: N18', 'TMenuItem', iptrw);
20750: ManualE1', 'TMenuItem', iptrw);
20751: FullTextFinder1', 'TMenuItem', iptrw);
20752: Move1', 'TMenuItem', iptrw);
20753: FractalDemo1', 'TMenuItem', iptrw);
20754: Tutorial21Android1', 'TMenuItem', iptrw);
20755: Tutorial0Function1', 'TMenuItem', iptrw);
20756: SimuLogBox1', 'TMenuItem', iptrw);
20757: OpenExamples1', 'TMenuItem', iptrw);
20758: SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
20759: JavaScriptSyntax1', 'TMenuItem', iptrw);
20760: Halt1', 'TMenuItem', iptrw);
20761: CodeSearch1', 'TMenuItem', iptrw);
20762: SynRubySyn1', 'TSynRubySyn', iptrw);
20763: RubySyntax1', 'TMenuItem', iptrw);
20764: Undo1', 'TMenuItem', iptrw);
20765: SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
20766: LinuxShellScript1', 'TMenuItem', iptrw);
20767: Rename1', 'TMenuItem', iptrw);
20768: spdcodesearch', 'TSpeedButton', iptrw);
20769: Preview1', 'TMenuItem', iptrw);
20770: Tutorial22Services1', 'TMenuItem', iptrw);
20771: Tutorial23RealTime1', 'TMenuItem', iptrw);
20772: Configuration1', 'TMenuItem', iptrw);
20773: MP3Player1', 'TMenuItem', iptrw);
20774: DLLSpy1', 'TMenuItem', iptrw);
20775: SynURIOpener1', 'TSynURIOpener', iptrw);
20776: SynURISyn1', 'TSynURISyn', iptrw);
20777: URILinksClicks1', 'TMenuItem', iptrw);
20778: EditReplace1', 'TMenuItem', iptrw);
20779: GotoLine1', 'TMenuItem', iptrw);
20780: ActiveLineColor1', 'TMenuItem', iptrw);
20781: ConfigFile1', 'TMenuItem', iptrw);
20782: SortIntlList', 'TMenuItem', iptrw);
20783: Redo1', 'TMenuItem', iptrw);
20784: Tutorial24CleanCode1', 'TMenuItem', iptrw);
20785: Tutorial25Configuration1', 'TMenuItem', iptrw);
20786: IndentSelection1', 'TMenuItem', iptrw);
```

```

20787: UnindentSection1', 'TMenuItem', iptrw);
20788: SkyStyle1', 'TMenuItem', iptrw);
20789: N19', 'TMenuItem', iptrw);
20790: CountWords1', 'TMenuItem', iptrw);
20791: imbookmarkimages', 'TImageList', iptrw);
20792: Bookmark11', 'TMenuItem', iptrw);
20793: N20', 'TMenuItem', iptrw);
20794: Bookmark21', 'TMenuItem', iptrw);
20795: Bookmark31', 'TMenuItem', iptrw);
20796: Bookmark41', 'TMenuItem', iptrw);
20797: SynMultiSyn1', 'TSynMultiSyn', iptrw);
20798:
20799: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
20800: Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSEExec; x:TPSRuntimeClassImporter);
20801: Procedure PSScriptCompile( Sender : TPSScript)
20802: Procedure Compile1Click( Sender : TObject)
20803: Procedure PSScriptExecute( Sender : TPSScript)
20804: Procedure open1Click( Sender : TObject)
20805: Procedure Save2Click( Sender : TObject)
20806: Procedure Savebefore1Click( Sender : TObject)
20807: Procedure Largefont1Click( Sender : TObject)
20808: Procedure FormActivate( Sender : TObject)
20809: Procedure SBytecode1Click( Sender : TObject)
20810: Procedure FormKeyPress( Sender : TObject; var Key : Char)
20811: Procedure Saveas3Click( Sender : TObject)
20812: Procedure Clear1Click( Sender : TObject)
20813: Procedure Slinenumbers1Click( Sender : TObject)
20814: Procedure About1Click( Sender : TObject)
20815: Procedure Search1Click( Sender : TObject)
20816: Procedure FormCreate( Sender : TObject)
20817: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
20818: var Action : TSynReplaceAction)
20819: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
20820: Procedure WordWrap1Click( Sender : TObject)
20821: Procedure SearchNext1Click( Sender : TObject)
20822: Procedure Replace1Click( Sender : TObject)
20823: Function PSScriptNeedFile(Sender:TObject;const OriginFileName:String;var FName,Output:String):Bool;
20824: Procedure ShowInclude1Click( Sender : TObject)
20825: Procedure Printout1Click( Sender : TObject)
20826: Procedure mnuPrintFont1Click( Sender : TObject)
20827: Procedure IncludelClick( Sender : TObject)
20828: Procedure FormDestroy( Sender : TObject)
20829: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
20830: Procedure UpdateView1Click( Sender : TObject)
20831: Procedure CodeCompletionList1Click( Sender : TObject)
20832: Procedure SaveOutput1Click( Sender : TObject)
20833: Procedure ExportClipboard1Click( Sender : TObject)
20834: Procedure Close1Click( Sender : TObject)
20835: Procedure Manual1Click( Sender : TObject)
20836: Procedure LoadLastFile1Click( Sender : TObject)
20837: Procedure Memo1Change( Sender : TObject)
20838: Procedure Decompile1Click( Sender : TObject)
20839: Procedure StepInto1Click( Sender : TObject)
20840: Procedure StepOut1Click( Sender : TObject)
20841: Procedure Reset1Click( Sender : TObject)
20842: Procedure cedebbugAfterExecute( Sender : TPSScript)
20843: Procedure cedebbugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
20844: Procedure cedebbugCompile( Sender : TPSScript)
20845: Procedure cedebbugExecute( Sender : TPSScript)
20846: Procedure cedebbugIdle( Sender : TObject)
20847: Procedure cedebbugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
20848: Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
20849: Procedure BreakPointMenuClick( Sender : TObject)
20850: Procedure DebugRun1Click( Sender : TObject)
20851: Procedure tutorial4Click( Sender : TObject)
20852: Procedure ImportfromClipboard1Click( Sender : TObject)
20853: Procedure ImportfromClipboard2Click( Sender : TObject)
20854: Procedure tutorial1Click( Sender : TObject)
20855: Procedure ShowSpecChars1Click( Sender : TObject)
20856: Procedure StatusBar1DblClick( Sender : TObject)
20857: Procedure PSScriptLine( Sender : TObject)
20858: Procedure Opendirectory1Click( Sender : TObject)
20859: Procedure procMessClick( Sender : TObject)
20860: Procedure tbtnUseCaseClick( Sender : TObject)
20861: Procedure EditFont1Click( Sender : TObject)
20862: Procedure tutorial21Click( Sender : TObject)
20863: Procedure tutorial31Click( Sender : TObject)
20864: Procedure HTMLEyntax1Click( Sender : TObject)
20865: Procedure ShowInterfaces1Click( Sender : TObject)
20866: Procedure Tutorial5Click( Sender : TObject)
20867: Procedure ShowLastException1Click( Sender : TObject)
20868: Procedure PlayMP31Click( Sender : TObject)
20869: Procedure AllFunctionsList1Click( Sender : TObject)
20870: Procedure texSyntax1Click( Sender : TObject)
20871: Procedure GetEMails1Click( Sender : TObject)
20872: procedure DelphiSite1Click(Sender: TObject);
20873: procedure TerminalStyle1Click(Sender: TObject);
20874: procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
20875: procedure ShellStyle1Click(Sender: TObject);

```

```
20876: procedure Console1Click(Sender: TObject); //3.2
20877: procedure BigScreen1Click(Sender: TObject);
20878: procedure Tutorial91Click(Sender: TObject);
20879: procedure SaveScreenshotClick(Sender: TObject);
20880: procedure Tutorial101Click(Sender: TObject);
20881: procedure SQLSyntax1Click(Sender: TObject);
20882: procedure XMLSyntax1Click(Sender: TObject);
20883: procedure ComponentCount1Click(Sender: TObject);
20884: procedure NewInstance1Click(Sender: TObject);
20885: procedure CSyntax1Click(Sender: TObject);
20886: procedure Tutorial6Click(Sender: TObject);
20887: procedure New1Click(Sender: TObject);
20888: procedure AllObjectsList1Click(Sender: TObject);
20889: procedure LoadBytecode1Click(Sender: TObject);
20890: procedure CipherFile1Click(Sender: TObject); //V3.5
20891: procedure NewInstance1Click(Sender: TObject);
20892: procedure toolbtnTutorialClick(Sender: TObject);
20893: procedure Memory1Click(Sender: TObject);
20894: procedure JavaSyntax1Click(Sender: TObject);
20895: procedure SyntaxCheck1Click(Sender: TObject);
20896: procedure ScriptExplorer1Click(Sender: TObject);
20897: procedure FormOutput1Click(Sender: TObject); //V3.6
20898: procedure GotoEnd1Click(Sender: TObject);
20899: procedure AllResourceList1Click(Sender: TObject);
20900: procedure tbtn6resClick(Sender: TObject); //V3.7
20901: procedure Info1Click(Sender: TObject);
20902: procedure Tutorial10Statistics1Click(Sender: TObject);
20903: procedure Tutorial11Forms1Click(Sender: TObject);
20904: procedure Tutorial12SQL1Click(Sender: TObject); //V3.8
20905: procedure ResourceExplore1Click(Sender: TObject);
20906: procedure Info1Click(Sender: TObject);
20907: procedure CryptoBox1Click(Sender: TObject);
20908: procedure ModulesCount1Click(Sender: TObject);
20909: procedure N4GewinntGame1Click(Sender: TObject);
20910: procedure PHPSyntax1Click(Sender: TObject);
20911: procedure SerialRS2321Click(Sender: TObject);
20912: procedure CSyntax2Click(Sender: TObject);
20913: procedure Calculator1Click(Sender: TObject);
20914: procedure Tutorial13Ciphering1Click(Sender: TObject);
20915: procedure Tutorial14Async1Click(Sender: TObject);
20916: procedure PHPSyntax1Click(Sender: TObject);
20917: procedure BtnZoomPlusClick(Sender: TObject);
20918: procedure BtnZoomMinusClick(Sender: TObject);
20919: procedure btnClassReportClick(Sender: TObject);
20920: procedure ThreadDemo1Click(Sender: TObject);
20921: procedure HEXView1Click(Sender: TObject);
20922: procedure ExporttoHTML1Click(Sender: TObject);
20923: procedure Minesweeper1Click(Sender: TObject);
20924: procedure PicturePuzzle1Click(Sender: TObject); //V3.9
20925: procedure sbvc1helpClick(Sender: TObject);
20926: procedure DependencyWalker1Click(Sender: TObject);
20927: procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
20928: procedure WebScanner1Click(Sender: TObject);
20929: procedure mnToolbar1Click(Sender: TObject);
20930: procedure mnStatusbar2Click(Sender: TObject);
20931: procedure mnConsole2Click(Sender: TObject);
20932: procedure mnCoolbar2Click(Sender: TObject);
20933: procedure mnSplitter2Click(Sender: TObject);
20934: procedure WebServer1Click(Sender: TObject);
20935: procedure PerlSyntax1Click(Sender: TObject);
20936: procedure PythonSyntax1Click(Sender: TObject);
20937: procedure DMathLibrary1Click(Sender: TObject);
20938: procedure IntfNavigator1Click(Sender: TObject);
20939: procedure FullTextFinder1Click(Sender: TObject);
20940: function AppName: string;
20941: function ScriptName: string;
20942: function LastName: string;
20943: procedure FractalDemo1Click(Sender: TObject);
20944: procedure SimuLogBox1Click(Sender: TObject);
20945: procedure OpenExamples1Click(Sender: TObject);
20946: procedure Halt1Click(Sender: TObject);
20947: procedure Stop;
20948: procedure CodeSearch1Click(Sender: TObject);
20949: procedure RubySyntax1Click(Sender: TObject);
20950: procedure Undo1Click(Sender: TObject);
20951: procedure LinuxShellScript1Click(Sender: TObject);
20952: procedure WebScannerDirect(urls: string);
20953: procedure WebScanner(urls: string);
20954: procedure LoadInterfaceList2;
20955: procedure DLLSpy1Click(Sender: TObject);
20956: procedure MemolDb1Click(Sender: TObject);
20957: procedure URILinksClicks1Click(Sender: TObject);
20958: procedure Gotoline1Click(Sender: TObject);
20959: procedure ConfigFile1Click(Sender: TObject);
20960: Procedure SortIntflistClick( Sender : TObject)
20961: Procedure Redo1Click( Sender : TObject)
20962: Procedure Tutorial24CleanCode1Click( Sender : TObject)
20963: Procedure IndentSelection1Click( Sender : TObject)
20964: Procedure UnindentSection1Click( Sender : TObject)
```

```

20965: Procedure SkyStyle1Click( Sender : TObject )
20966: Procedure CountWords1Click( Sender : TObject )
20967: Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark )
20968: Procedure Memo1GutterClick( Sender : TObject; Button : TMouseButton; X, Y, Line : Integer; Mark : TSynEditMark );
20969: Procedure Bookmark11Click( Sender : TObject )
20970: Procedure Bookmark21Click( Sender : TObject )
20971: Procedure Bookmark31Click( Sender : TObject )
20972: Procedure Bookmark41Click( Sender : TObject )
20973: Procedure SynMultiSyn1CustomRange( Sender : TSynMultiSyn; Operat : TRangeOperation; var Range : Pointer );
20974: 'STATMemoryReport', 'boolean', iptrw );
20975: 'IPPort', 'integer', iptrw );
20976: 'COMPort', 'integer', iptrw );
20977: 'lbintlist', 'TListBox', iptrw );
20978: Function GetStatChange : boolean
20979: Procedure SetStatChange( vstat : boolean )
20980: Function GetActFileName : string
20981: Procedure SetActFileName( vname : string )
20982: Function GetLastFileName : string
20983: Procedure SetLastFileName( vname : string )
20984: Procedure WebScannerDirect( urls : string )
20985: Procedure LoadInterfaceList2
20986: Function GetStatExecuteShell : boolean
20987: Procedure DoEditorExecuteCommand( EditorCommand : word )
20988: function GetActiveLineColor: TColor
20989: procedure SetActiveLineColor( acolor: TColor )
20990: procedure ScriptListbox1Click( Sender : TObject );
20991: procedure Memo2KeyPress( Sender : TObject; var Key : Char );
20992: procedure EnlargeGutter1Click( Sender : TObject );
20993: procedure Tetris1Click( Sender : TObject );
20994: procedure ToDoList1Click( Sender : TObject );
20995: procedure ProcessList1Click( Sender : TObject );
20996: procedure MetricReport1Click( Sender : TObject );
20997: procedure ProcessList1Click( Sender : TObject );
20998: procedure TCPSockets1Click( Sender : TObject );
20999: procedure ConfigUpdate1Click( Sender : TObject );
21000: procedure ADOWorkbench1Click( Sender : TObject );
21001: procedure SocketServer1Click( Sender : TObject );
21002: procedure FormDemo1Click( Sender : TObject );
21003: procedure Richedit1Click( Sender : TObject );
21004: procedure SimpleBrowser1Click( Sender : TObject );
21005: procedure DOSShell1Click( Sender : TObject );
21006: procedure SynExport1Click( Sender : TObject );
21007: procedure ExporttoRTF1Click( Sender : TObject );
21008: procedure FormCloseQuery( Sender : TObject; var CanClose : Boolean );
21009: procedure SOAPTester1Click( Sender : TObject );
21010: procedure Sniffer1Click( Sender : TObject );
21011: procedure AutoDetectSyntax1Click( Sender : TObject );
21012: procedure FPlot1Click( Sender : TObject );
21013: procedure PasStyle1Click( Sender : TObject );
21014: procedure Tutorial183RGBLED1Click( Sender : TObject );
21015: procedure ReversilClick( Sender : TObject );
21016: procedure Manualmaxbox1Click( Sender : TObject );
21017: procedure BlaisePascalMagazine1Click( Sender : TObject );
21018: procedure AddToDo1Click( Sender : TObject );
21019: procedure CreateGUID1Click( Sender : TObject );
21020: procedure Tutorial27XML1Click( Sender : TObject );
21021: procedure CreateDLLStub1Click( Sender : TObject );
21022: procedure Tutorial28DLL1Click( Sender : TObject );'
21023: procedure ResetKeyPressed();'
21024: procedure KeyPressedFalse;
21025: procedure FileChanges1Click( Sender : TObject );'
21026: procedure OpenGLTry1Click( Sender : TObject );'
21027: procedure AllUnitList1Click( Sender : TObject );'
21028: procedure Tutorial29UMLClick( Sender : TObject );
21029: procedure CreateHeader1Click( Sender : TObject );
21030: procedure Oscilloscope1Click( Sender : TObject );'
21031: procedure Tutorial30WOT1Click( Sender : TObject );'
21032: procedure GetWebScript1Click( Sender : TObject );'
21033: procedure Checkers1Click( Sender : TObject );'
21034: procedure TaskMgr1Click( Sender : TObject );'
21035: procedure WebCam1Click( Sender : TObject );'
21036: procedure Tutorial31Closure1Click( Sender : TObject );'
21037: procedure GEOMapView1Click( Sender : TObject );'
21038: procedure Run1Click( Sender : TObject );
21039: MaxForm1.GPSSatView1Click, 'GPSSatView1Click');
21040: MaxForm1.N3DLab1Click, 'N3DLab1Click');
21041:
21042: //-----
21043: //*****mX4 Editor SynEdit Tools API *****
21044: //-----
21045: procedure SIRegister_TCustomSynEdit( CL : TPPSPascalCompiler );
21046: begin
21047: //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
21048: with FindClass('TCustomControl','TCustomSynEdit') do begin
21049:   Constructor Create( AOwner : TComponent )
21050:   SelStart', 'Integer', iptrw );
21051:   SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw );
21052:   Procedure UpdateCaret
21053:   Procedure AddKey(Command: TSynEditorCommand; Key1:word; SS1:TShiftState; Key2:word; SS2:TShiftState);

```

```

21054: Procedure AddKey(Command: TSynEditorCommand; Key1: word; SS1:TShiftState; Key2: word; SS2:TShiftState);
21055: Procedure BeginUndoBlock
21056: Procedure BeginUpdate
21057: Function CaretInView : Boolean
21058: Function CharIndexToRowCol( Index : integer ) : TBufferCoord
21059: Procedure Clear
21060: Procedure ClearAll
21061: Procedure ClearBookMark( BookMark : Integer )
21062: Procedure ClearSelection
21063: Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer )
21064: Procedure ClearUndo
21065: Procedure CopyToClipboard
21066: Procedure CutToClipboard
21067: Procedure DoCopyToClipboard( const SText : string )
21068: Procedure EndUndoBlock
21069: Procedure EndUpdate
21070: Procedure EnsureCursorPosVisible
21071: Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean )
21072: Procedure FindMatchingBracket
21073: Function GetMatchingBracket : TBufferCoord
21074: Function GetMatchingBracketEx( const APoint : TBufferCoord ) : TBufferCoord
21075: Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer )
21076: Function GetBookMark( BookMark : integer; var X, Y : integer ) : boolean
21077: Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attr : TSynHighlighterAttributes ) : boolean
21078: Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
21079: var TokenType, Start : Integer; var Attr:TSynHighlighterAttributes):boolean
21080: Function GetPositionOfMouse( out aPos : TBufferCoord ) : Boolean
21081: Function GetWordAtRowCol( const XY : TBufferCoord ) : string
21082: Procedure GotoBookMark( BookMark : Integer )
21083: Procedure GotoLineAndCenter( ALine : Integer )
21084: Function IdentChars : TSynIdentChars
21085: Procedure InvalidateGutter
21086: Procedure InvalidateGutterLine( aLine : integer )
21087: Procedure InvalidateGutterLines( FirstLine, LastLine : integer )
21088: Procedure InvalidateLine( Line : integer )
21089: Procedure InvalidateLines( FirstLine, LastLine : integer )
21090: Procedure InvalidateSelection
21091: Function IsBookmark( BookMark : integer ) : boolean
21092: Function IsPointInSelection( const Value : TBufferCoord ) : boolean
21093: Procedure LockUndo
21094: Function BufferToDisplayPos( const p : TBufferCoord ) : TDisplayCoord
21095: Function DisplayToBufferPos( const p : TDisplayCoord ) : TBufferCoord
21096: Function LineToRow( aLine : integer ) : integer
21097: Function RowToLine( aRow : integer ) : integer
21098: Function NextWordPos : TBufferCoord
21099: Function NextWordPosEx( const XY : TBufferCoord ) : TBufferCoord
21100: Procedure PasteFromClipboard
21101: Function WordStart : TBufferCoord
21102: Function WordStartEx( const XY : TBufferCoord ) : TBufferCoord
21103: Function WordEnd : TBufferCoord
21104: Function WordEndEx( const XY : TBufferCoord ) : TBufferCoord
21105: Function PrevWordPos : TBufferCoord
21106: Function PrevWordPosEx( const XY : TBufferCoord ) : TBufferCoord
21107: Function PixelsToRowColumn( aX, aY : integer ) : TDisplayCoord
21108: Function PixelsToNearestRowColumn( aX, aY : integer ) : TDisplayCoord
21109: Procedure Redo
21110: Procedure RegisterCommandHandler( const AHandlerProc:THookedCommandEvent; AHandlerData:pointer );
21111: Function RowColumnToPixels( const RowCol : TDisplayCoord ) : TPoint
21112: Function RowColToCharIndex( RowCol : TBufferCoord ) : integer
21113: Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
21114: Procedure SelectAll
21115: Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer )
21116: Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord )
21117: Procedure SetDefaultKeystrokes
21118: Procedure SetSelWord
21119: Procedure SetWordBlock( Value : TBufferCoord )
21120: Procedure Undo
21121: Procedure UnlockUndo
21122: Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent )
21123: Procedure UnRegisterKeyUpHandler( aHandler : TKeyEvent )
21124: Procedure AddKeyUpHandler( aHandler : TKeyEvent )
21125: Procedure RemoveKeyUpHandler( aHandler : TKeyEvent )
21126: Procedure AddKeyDownHandler( aHandler : TKeyEvent )
21127: Procedure RemoveKeyDownHandler( aHandler : TKeyEvent )
21128: Procedure AddKeyPressHandler( aHandler : TKeyPressEvent )
21129: Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent )
21130: Procedure AddFocusControl( aControl : TWInControl )
21131: Procedure RemoveFocusControl( aControl : TWInControl )
21132: Procedure AddMouseDownHandler( aHandler : TMouseEvent )
21133: Procedure RemoveMouseDownHandler( aHandler : TMouseEvent )
21134: Procedure AddMouseUpHandler( aHandler : TMouseEvent )
21135: Procedure RemoveMouseUpHandler( aHandler : TMouseEvent )
21136: Procedure AddMouseCursorHandler( aHandler : TMouseEvent )
21137: Procedure RemoveMouseCursorHandler( aHandler : TMouseEvent )
21138: Procedure SetLinesPointer( ASynEdit : TCustomSynEdit )
21139: Procedure RemoveLinesPointer
21140: Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList )
21141: Procedure UnHookTextBuffer
21142: BlockBegin', 'TBufferCoord', iptrw);

```

```

21143:     BlockEnd', 'TBufferCoord', iptrw);
21144:     CanPaste', 'Boolean', iptr);
21145:     CanRedo', 'boolean', iptr);
21146:     CanUndo', 'boolean', iptr);
21147:     CaretX', 'Integer', iptrw);
21148:     CaretY', 'Integer', iptrw);
21149:     CaretXY', 'TBufferCoord', iptrw);
21150:     ActiveLineColor', 'TColor', iptrw);
21151:     DisplayX', 'Integer', iptr);
21152:     DisplayY', 'Integer', iptr);
21153:     DisplayXY', 'TDisplayCoord', iptr);
21154:     DisplayLineCount', 'integer', iptr);
21155:     CharsInWindow', 'Integer', iptr);
21156:     CharWidth', 'integer', iptr);
21157:     Font', 'TFont', iptrw);
21158:     GutterWidth', 'Integer', iptr);
21159:     Highlighter', 'TSynCustomHighlighter', iptrw);
21160:     LeftChar', 'Integer', iptrw);
21161:     LineHeight', 'integer', iptr);
21162:     LinesInWindow', 'Integer', iptr);
21163:     LineText', 'string', iptrw);
21164:     Lines', 'TStrings', iptrw);
21165:     Marks', 'TSynEditMarkList', iptr);
21166:     MaxScrollWidth', 'integer', iptrw);
21167:     Modified', 'Boolean', iptrw);
21168:     PaintLock', 'Integer', iptr);
21169:     ReadOnly', 'Boolean', iptrw);
21170:     SearchEngine', 'TSynEditSearchCustom', iptrw);
21171:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
21172:     SelTabBlock', 'Boolean', iptr);
21173:     SelTabLine', 'Boolean', iptr);
21174:     SelText', 'string', iptrw);
21175:     StateFlags', 'TSynStateFlags', iptr);
21176:     Text', 'string', iptrw);
21177:     TopLine', 'Integer', iptrw);
21178:     WordAtCursor', 'string', iptr);
21179:     WordAtMouse', 'string', iptr);
21180:     UndoList', 'TSynEditUndoList', iptr);
21181:     RedoList', 'TSynEditUndoList', iptr);
21182:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
21183:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
21184:     BorderStyle', 'TSynBorderStyle', iptrw);
21185:     ExtraLineSpacing', 'integer', iptrw);
21186:     Gutter', 'TSynGutter', iptrw);
21187:     HideSelection', 'boolean', iptrw);
21188:     InsertCaret', 'TSynEditCaretType', iptrw);
21189:     InsertMode', 'boolean', iptrw);
21190:     IsScrolling', 'Boolean', iptr);
21191:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
21192:     MaxUndo', 'Integer', iptrw);
21193:     Options', 'TSynEditorOptions', iptrw);
21194:     OverwriteCaret', 'TSynEditCaretType', iptrw);
21195:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
21196:     ScrollHintColor', 'TColor', iptrw);
21197:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
21198:     ScrollBars', 'TScrollStyle', iptrw);
21199:     SelectedColor', 'TSynSelectedColor', iptrw);
21200:     SelectionMode', 'TSynSelectionMode', iptrw);
21201:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
21202:     TabWidth', 'integer', iptrw); WantReturns', 'boolean', iptrw);
21203:     WantTabs', 'boolean', iptrw); WordWrap', 'boolean', iptrw);
21204:     WordWrapGlyph', 'TSynGlyph', iptrw);
21205:     OnChange', 'TNotifyEvent', iptrw);
21206:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
21207:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
21208:     OnContextHelp', 'TContextHelpEvent', iptrw);
21209:     OnDropFiles', 'TDropFilesEvent', iptrw);
21210:     OnGutterClick', 'TGutterClickEvent', iptrw);
21211:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
21212:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
21213:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
21214:     OnPaint', 'TPaintEvent', iptrw);
21215:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
21216:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
21217:     OnReplaceText', 'TReplaceTextEvent', iptrw);
21218:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
21219:     OnStatusChange', 'TStatusChangeEvent', iptrw);
21220:     OnPaintTransient', 'TPaintTransient', iptrw);
21221:     OnScroll', 'TScrollEvent', iptrw);
21222:   end;
21223:   Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
21224:   Function GetPlaceableHighlighters : TSynHighlighterList
21225:   Function EditorCommandToDescrString( Cmd : TSynEditorCommand ) : string
21226:   Function EditorCommandToCodeString( Cmd : TSynEditorCommand ) : string
21227:   Procedure GetEditorCommandValues( Proc : TGetStrProc )
21228:   Procedure GetEditorCommandExtended( Proc : TGetStrProc )
21229:   Function IdentToEditorCommand( const Ident : string; var Cmd : longint ) : boolean
21230:   Function EditorCommandToIdent( Cmd : longint; var Ident : string ) : boolean
21231:   Function ConvertCodeStringToExtended( AString : String ) : String

```

```

21232: Function ConvertExtendedToCodeString( AString : String ) : String
21233: Function ConvertExtendedToCommand( AString : String ) : TSynEditorCommand
21234: Function ConvertCodeStringToCommand( AString : String ) : TSynEditorCommand
21235: Function IndexToEditorCommand( const AIndex : Integer ) : Integer
21236:
21237: TSynEditorOption = (
21238:   eoAltSetsColumnMode,           //Holding down the Alt Key will put the selection mode into columnar format
21239:   eoAutoIndent,                 //Will indent caret on newlines with same amount of leading whitespace as
21240:                           // preceding line
21241:   eoAutoSizeMaxScrollWidth,     //Automatically resizes the MaxScrollWidth property when inserting text
21242:   eoDisableScrollArrows,       //Disables the scroll bar arrow buttons when you can't scroll in that
21243:                           //direction any more
21244:   eoDragDropEditing,          //Allows to select a textblock and drag it in document to another location
21245:   eoDropFiles,                //Allows the editor accept OLE file drops
21246:   eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
21247:   eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
21248:   eoGroupUndo,               //When undoing/redoing actions, handle all cont.changes same kind in onecall
21249:                           //instead undoing/redoing each command separately
21250:   eoHalfPageScroll,          //By scrolling with page-up/page-down commands, only scroll half page attime
21251:   eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
21252: If you have ScrollPasteEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
21253:   eoKeepCaretX,             //When moving through lines w/o cursor Past EOL, keeps X position of cursor
21254:   eoNoCaret,                //Makes it so the caret is never visible
21255:   eoNoSelection,            //Disables selecting text
21256:   eoRightMouseMovesCursor, //When clicking with right mouse for popup menu, moves cursor to location
21257:   eoScrollByOneLess,        //Forces scrolling to be one less
21258:   eoScrollHintFollows,      //The scroll hint follows the mouse when scrolling vertically
21259:   eoScrollPastEof,          //Allows the cursor to go past the end of file marker
21260:   eoScrollPastEol,          //Allows cursor to go past last character into white space at end of a line
21261:   eoShowScrollHint,         //Shows a hint of the visible line numbers when scrolling vertically
21262:   eoShowSpecialChars,       //Shows the special Characters
21263:   eoSmartTabDelete,         //similar to Smart Tabs, but when you delete characters
21264:   eoSmartTabs,              //When tabbing, cursor will go to non-white space character of previous line
21265:   eoSpecialLineDefaultFg,   //disables the foreground text color override using OnSpecialLineColor event
21266:   eoTabIndent,              //If active <Tab>and<Shift><Tab> act block indent, unindent when text select
21267:   eoTabsToSpaces,           //Converts a tab character to a specified number of space characters
21268:   eoTrimTrailingSpaces     //Spaces at the end of lines will be trimmed and not saved
21269:
21270: *****Important Editor Short Cuts*****
21271: Double click to select a word and count words with highlighting.
21272: Triple click to select a line.
21273: CTRL+SHIFT+click to extend a selection.
21274: Drag with the ALT key down to select columns of text !!!
21275: Drag and drop is supported.
21276: Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
21277: Type CTRL+A to select all.
21278: Type CTRL+N to set a new line.
21279: Type CTRL+T to delete a line or token. //Tokenizer
21280: Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
21281: Type CTRL+Shift+T to add ToDo in line and list.
21282: Type CTRL+Shift+[0..9] to set bookmarks. //Bookmark
21283: Type CTRL[0..9] to jump or get to bookmarks.
21284: Type Home to position cursor at beginning of current line and End to position it at end of line.
21285: Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
21286: Page Up and Page Down work as expected.
21287: CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
21288: using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
21289:
21290: {$ Short Key Positions Ctrl<A-Z>: }
21291: def
21292:   <A> Select All
21293:   <B> Count Words
21294:   <C> Copy
21295:   <D> Internet Start
21296:   <E> Script List
21297:   <F> Find
21298:   <G> Goto
21299:   <H> Mark Line
21300:   <I> Interface List
21301:   <J> Code Completion
21302:   <K> Console
21303:   <L> Interface List Box
21304:   <M> Font Larger -
21305:   <N> New Line
21306:   <O> Open File
21307:   <P> Font Smaller +
21308:   <Q> Quit
21309:   <R> Replace
21310:   <S> Save!
21311:   <T> Delete Line
21312:   <U> Use Case Editor
21313:   <V> Paste
21314:   <W> URI Links
21315:   <X> Reserved for coding use internal
21316:   <Y> Delete Line
21317:   <Z> Undo
21318:
21319: ref
21320: F1 Help

```

```

21321: F2 Syntax Check
21322: F3 Search Next
21323: F4 New Instance
21324: F5 Line Mark /Breakpoint
21325: F6 Goto End
21326: F7 Debug Step Into
21327: F8 Debug Step Out
21328: F9 Compile
21329: F10 Menu
21330: F11 Word Count Highlight
21331: F12 Reserved for coding use internal
21332:
21333: AddRegisteredVariable('Application', 'TApplication');
21334: AddRegisteredVariable('Screen', 'TScreen');
21335: AddRegisteredVariable('Self', 'TForm');
21336: AddRegisteredVariable('Memo1', 'TSynMemo');
21337: AddRegisteredVariable('memo2', 'TMemo');
21338: AddRegisteredVariable('maxForm1', 'TMaxform1'); //!
21339: AddRegisteredVariable('debugout', 'Tdebugoutput'); //!
21340: AddRegisteredVariable('hlog', 'THotlog'); //!
21341: AddRegisteredVariable(it, integer); //for closure!
21342: AddRegisteredVariable(sr, string); //for closure
21343: AddRegisteredVariable(bt, boolean); //for closure
21344: AddRegisteredVariable(ft, double); //for closure
21345: AddRegisteredVariable(srlst, TStringlist); //for closures
21346:
21347: def ReservedWords: array[0..82] of string =
21348:   ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
21349:    'constructor', 'default', 'destructor', 'disinterface', 'div', 'do',
21350:    'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
21351:    'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
21352:    'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
21353:    'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
21354:    'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
21355:    'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
21356:    'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
21357:    'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
21358:    'public', 'published', def, ref, using, typedef, memo1, 'memo2', 'doc', 'maxform1', 'it';
21359: AllowedChars: array[0..5] of string = ('(,),', '[, ]', ' ', t,t1,t2,t3: boolean);
21360: //-----
21361: //*****End of mx4 Public Tools API *****
21362: //-----
21363:
21364: Amount of Functions: 13694
21365: Amount of Procedures: 8406
21366: Amount of Constructors: 1369
21367: Totals of Calls: 23468
21368: SHA1: Win 3.9.9.101 E951E93E67DE0E815D3F7441842994BAEA06F21A
21369:
21370: ****
21371: Doc Short Manual with 50 Tips!
21372: ****
21373: - Install: just save your maxboxdef.ini before and then extract the zip file!
21374: - Toolbar: Click on the red maxbox Sign (right on top) opens your work directory or jump to <Help>
21375: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
21376: - Menu: With <Ctrl><F3> you can search for code on examples
21377: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
21378: - Menu: Set Interface Navigator in menu /View/Intf Navigator
21379: - Menu: Switch or toggle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
21380:
21381: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
21382: - Inifile: Refresh (reload) the inifile after edit with ..\Help\Config Update
21383: - Context Menu: You can printout your scripts as a pdf-file or html-export
21384: - Context: You do have a context menu with the right mouse click
21385:
21386: - Menu: With the UseCase Editor you can convert graphic formats too.
21387: - Menu: On menu Options you find Addons as compiled scripts
21388: - IDE: Menu Program: Run Only is faster, after F2 - You don't need a mouse use shortcuts
21389: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
21390: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
21391: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or interface
21392:   or ttimer (you type classp and then CTRL J), or you type tstringlist and <Ctrl><J>
21393:
21394: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
21395: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
21396: - Code: If you code a loop till key-pressed use function: isKeyPressed;
21397: - Code: Macro set the macros #name,, #paAdministrator, #file,startmaxbox_extract_funcList399.txt
21398: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
21399: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
21400:   to delete and Click and mark to drag a bookmark
21401: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
21402: - IDE: A file info with system and script information you find in menu Program/Information
21403: - IDE: After change the config file in help you can update changes in menu Help/Config Update
21404: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
21405: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
21406: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
21407: - Editor: Set Bookmarks to check your work in app or code
21408: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
21409: - Editor: With //TODO: some description or DONE you set code entries for ToDo List in ..\Help/ToDo List

```

```

21410: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
21411:
21412: - IDE with menu /Options/ADO SQL Workbench you can manage your Database
21413: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
21414: - Menu: Set Interface Naviagator also with toggle <Ctrl L> or /View/Inf Navigator
21415: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
21416: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
21417: - Code Editor: Compile with <F9> but also Alt C in case <F9> isn't available;
21418: - IDE set bookmarks with <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
21419: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
21420: - IDE menu /Help/Tools/ open the Task Manager
21421:
21422: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
21423: - Add on when no browser is available start /Options/Add ons/Easy Browser
21424: - Add on SOAP Tester with SOP POST File
21425: - Add on IP Protocol Sniffer with List View
21426: - Add on OpenGL mX Robot Demo for android
21427: - Add on Checkers Game, Add on Oscilloscope
21428:
21429: - Menu: Help/Tools as a Tool Section with DOS Opener
21430: - Menu Editor: export the code as RTF File
21431: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
21432: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
21433: - Context: Auto Detect of Syntax depending on file extension
21434: - Code: some Windows API function start with w in the name like wGetAtomName();
21435: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
21436: - IDE File Check with menu ..View/File Changes/...
21437: - Context: Create a Header with Create Header in Navigator List at right window
21438: - Code: use SysErrorMessage to get a real Error Description, Ex.
21439:     RemoveDir('c:\NoSuchFolder');
21440:     writeln('System Error Message: '+ SysErrorMessage(GetLastError));
21441: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
21442:
21443: - using DLL example in maXbox: //function: {*****}
21444: Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
21445:                                     cb: DWORD): BOOL; //stdcall;
21446: External 'GetProcessMemoryInfo@psapi.dll stdcall';
21447: Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
21448: External 'OpenProcess@kernel32.dll stdcall';
21449:
21450: GCC Compile Ex Script
21451: procedure TFormMain_btnCompileClick(Sender: TObject);
21452: begin
21453:   AProcess:= TProcess.Create(nil);
21454:   try
21455:     AProcess.Commandline := 'gcc.exe "' + OpenDialog1.FileName + '"'
21456:     +' -O "' + OpenDialog2.FileName + '"';
21457:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
21458:     AProcess.Execute;
21459:     Memo2.Lines.BeginUpdate;
21460:     Memo2.Lines.Clear;
21461:     Memo2.Lines.LoadFromStream(AProcess.Output);
21462:     Memo2.Lines.EndUpdate;
21463:   finally
21464:     AProcess.Free;
21465:   end;
21466:
21467: Stopwatch pattern
21468: Time1:= Time;
21469: writeln(formatdatetime('start: hh:mm:ss:zzz',Time))
21470: if initAndStartBoard then
21471:   writeln('Filesize: '+inttostr(filesize(FILESAVE)));
21472: writeln(formatDateTime('stop: hh:mm:ss:zzz',Time))
21473: PrintF('%d %s',[Trunc((Time-Time1)*24),
21474:                   FormatDateTime('h runtime: nn:ss:zzz',Time-Time1)])
21475:
21476: POST git-receive-pack (chunked)
21477: Pushing to https://github.com/maxkleiner/maXbox3.git
21478: To https://github.com/maxkleiner/maXbox3.git f127d21..c6a98da masterbox2 -> masterbox2
21479: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
21480:
21481: History Shell Hell
21482: PCT Precompile Technology , mX4 ScriptStudio
21483: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
21484: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
21485: emax layers: system-package-component-unit-class-function-block
21486: new keywords def ref using maXCalcF
21487: UML: use case act class state seq pac comp dep - lib lab
21488: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
21489: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
21490: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
21491: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
21492: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
21493: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
21494: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
21495: SendBuffer, Color+Caption hack, ComboSet, SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
21496: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
21497: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
21498: TFixedCriticalSection, Xplatform beta, GCC Command Pipe

```

```

21499: Inno Install and Setup Routines
21500: Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
21501: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21502: Vfw (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
21503: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
21504: Add 5 Units, 1 Tutors, maxmap, OpenStreetView, MAPX
21505: Function Menu/View/GEO Map View, DownloadFile, wgetX, sensors
21506: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
21507: ByteCode2, IPUtils2, GEOCode, CGI-Powtils, GPS_2
21508:
21509: Ref:
21510: https://unibe-ch.academia.edu/MaxKleiner
21511: http://www.slideshare.net/maxkleiner1
21512: http://www.scribd.com/max_kleiner
21513: http://www.delphiforfun.org/Programs/Utilities/index.htm
21514: http://www.slideshare.net/maxkleiner1
21515: http://s3.amazonaws.com/PreviewLinks/22959.html
21516: http://www.softwareschule.ch/arduino_training.pdf
21517: http://www.jrsoftware.org/isinfo.php
21518: http://www.be-precision.com/products/precision-builder/express/
21519: http://www.blaisepascal.eu/
21520: http://www.delphibasics.co.uk/
21521: http://www.youtube.com/watch?v=av89HAbqAsI
21522: http://www.angelfire.com/his5/delphizeus/modal.html
21523: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
21524: http://delphi.org/2014/01/every-android-api-for-delphi/
21525: https://en.wikipedia.org/wiki/User:Maxkleiner
21526: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
21527:
21528: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chl=%s';
21529: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
21530: =renderBasicSearchNarrative&q=%s';
21531: UrlMapQuestAPIReverse='http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
21532: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
21533:
21534:
21535: function OpenMap(const Data: string): boolean;
21536: var encURL: string;
21537: begin
21538:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPENcode(Data)]);
21539:   try
21540:     //HttpGet(EncodedURL, mapStream); //WinInet
21541:     Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
21542:     //OpenDoc(Exepath+'openmapx.html');
21543:     S_ShellExecute(Exepath+'openmapx.html','','seCmdOpen');
21544:   finally
21545:     encURL:= '';
21546:   end;
21547: end;
21548:
21549: procedure GetGEOMap(C_form,apath: string; const Data: string);
21550: var
21551:   encodedURL: string;
21552:   mapStream: TMemoryStream;
21553: begin
21554:   //encodedURL:= Format(UrlGoogleQrCode,[Width,Height, C_Level, HTTPENcode(Data)]);
21555:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPENcode(Data)]);
21556:   mapStream:= TMemoryStream.create;
21557:   try
21558:     Wininet_HttpGet(EncodedURL, mapStream); //WinInet
21559:     mapStream.Position:= 0;
21560:     mapStream.Savetofile(apath); // OpenDoc(apath);
21561:     S_ShellExecute(apath,'',seCmdOpen);
21562:   finally
21563:     mapStream.Free;
21564:   end;
21565: end;
21566:
21567:
21568: ****
21569: unit List asm internal end
21570: ****
21571: 01 unit RIRegister_Utils_Routines(exec); //Delphi
21572: 02 unit SIRegister_IdStrings; //Indy Sockets
21573: 03 unit RIRegister_niSTRING_Routines(Exec); //from RegEx
21574: 04 unit uPSI_fMain Functions; //maxbox Open Tools API
21575: 05 unit IFSI_WinForm1puzzle; //maxbox
21576: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
21577: 07 unit RegisterDateTimelibrary_R(exec); //Delphi
21578: 08 unit RIRegister_MathMax_Routines(exec); //Jedi & Delphi
21579: 09 unit RIRegister_IdGlobal_Routines(exec); //Indy Sockets
21580: 10 unit RIRegister_SysUtils_Routines(Exec); //Delphi
21581: 11 unit uPSI_IdTCPConnection; //Indy some functions
21582: 12 unit uPSCompiler.pas; //PS kernel functions
21583: 13 unit uPSI_DBCommon; //DB Common_Routines and Types
21584: 14 unit uPSI_Printers.pas; //Delphi VCL
21585: 15 unit uPSI_MPlayer.pas; //Delphi VCL
21586: 16 unit uPSC_comobj; //COM Functions
21587: 17 unit uPSI_Clipbrd; //Delphi VCL

```

```

21588: 18 unit Filectrl in IFSI_SysUtils_max;           //VCL Runtime
21589: 19 unit uPSI_SqlExpr;                          //DBX3
21590: 20 unit uPSI_ADOdb;                           //ADODB
21591: 21 unit uPSI_StrHlpr;                         //String Helper Routines
21592: 22 unit uPSI_DateUtils;                        //Expansion to Date/TimeLib
21593: 23 unit uPSI_FileUtils;                        //Expansion to Sys/File Utils
21594: 24 unit JUutils / gsUtils;                   //Jedi / Metabase
21595: 25 unit JvFunctions_max;                      //Jedi Functions
21596: 26 unit HTTPParser;                           //Delphi VCL
21597: 27 unit HTTPUtil;                            //Delphi VCL
21598: 28 unit uPSI_XMLUtil;                         //Delphi VCL
21599: 29 unit uPSI_SOAPHTTPClient;                 //Delphi VCL SOAP WebService V3.5
21600: 30 unit uPSI_Contnrs;                         //Delphi RTL Container of Classes
21601: 31 unit uPSI_MaskUtils;                       //RTL Edit and Mask functions
21602: 32 unit uPSI_MyBigInt;                        //big integer class with Math
21603: 33 unit uPSI_ConvUtils;                       //Delphi VCL Conversions engine
21604: 34 unit Types_Variants;                      //Delphi Win32\rtl\sys
21605: 35 unit uPSI_IdHashSHA1;                     //Indy Crypto Lib
21606: 36 unit uPSI_IdHashMessageDigest;             //Indy Crypto;
21607: 37 unit uPSI_IdASN1Util;                     //Indy ASN1Utility Routines;
21608: 38 unit uPSI_IdLogFile;                       //Indy Logger from LogBase
21609: 39 unit uPSI_IdICmpClient;                   //Indy Ping ICMP
21610: 40 unit uPSI_IdHashMessageDigest_max;        //Indy Crypto &OpenSSL;
21611: 41 unit uPSI_FileCtrl;                        //Delphi RTL
21612: 42 unit uPSI_Outline;                         //Delphi VCL
21613: 43 unit uPSI_ScktComp;                        //Delphi RTL
21614: 44 unit uPSI_Calendar;                       //Delphi VCL
21615: 45 unit uPSI_VlistView;                      //VListView;
21616: 46 unit uPSI_DBGrids;                         //Delphi VCL
21617: 47 unit uPSI_DBCtrls;                        //Delphi VCL
21618: 48 unit ide_debugoutput;                     //maxbox
21619: 49 unit uPSI_ComCtrls;                       //Delphi VCL
21620: 50 unit uPSC_stdCtrls+;                      //Delphi VCL
21621: 51 unit uPSI_Dialogs;                        //Delphi VCL
21622: 52 unit uPSI_StdConvs;                       //Delphi RTL
21623: 53 unit uPSI_DBClient;                       //Delphi RTL
21624: 54 unit uPSI_DBPlatform;                     //Delphi RTL
21625: 55 unit uPSI_Provider;                       //Delphi RTL
21626: 56 unit uPSI_FMTBcd;                         //Delphi RTL
21627: 57 unit uPSI_DBCGrids;                       //Delphi VCL
21628: 58 unit uPSI_CDSSUtil;                       //MIDAS
21629: 59 unit uPSI_VarHlpr;                        //Delphi RTL
21630: 60 unit uPSI_ExtDlg;                         //Delphi VCL
21631: 61 unit sdpStopwatch;                       //maxbox
21632: 62 unit uPSI_JclStatistics;                 //JCL
21633: 63 unit uPSI_JclLogic;                       //JCL
21634: 64 unit uPSI_JclMiscel;                      //JCL
21635: 65 unit uPSI_JclMath_max;                    //JCL RTL
21636: 66 unit uPSI_uTPLb_StreamUtils;              //LockBox 3
21637: 67 unit uPSI_MathUtils;                      //BCB
21638: 68 unit uPSI_JclMultimedia;                 //JCL
21639: 69 unit uPSI_WideStrUtils;                  //Delphi API/RTL
21640: 70 unit uPSI_GraphUtil;                      //Delphi RTL
21641: 71 unit uPSI_TypeTrans;                     //Delphi RTL
21642: 72 unit uPSI_HTTPApp;                        //Delphi VCL
21643: 73 unit uPSI_DBWeb;                          //Delphi VCL
21644: 74 unit uPSI_DBBdeWeb;                       //Delphi VCL
21645: 75 unit uPSI_DBXpressWeb;                   //Delphi VCL
21646: 76 unit uPSI_ShadowWnd;                     //Delphi VCL
21647: 77 unit uPSI_ToolWin;                        //Delphi VCL
21648: 78 unit uPSI_Tabs;                           //Delphi VCL
21649: 79 unit uPSI_JclGraphUtils;                 //JCL
21650: 80 unit uPSI_JclCounter;                    //JCL
21651: 81 unit uPSI_JclSysInfo;                   //JCL
21652: 82 unit uPSI_JclSecurity;                  //JCL
21653: 83 unit uPSI_JclFileUtils;                 //JCL
21654: 84 unit uPSI_IdUserAccounts;                //Indy
21655: 85 unit uPSI_IdAuthentication;              //Indy
21656: 86 unit uPSI_uTPLb_AES;                     //LockBox 3
21657: 87 unit uPSI_IdHashSHA1;                   //LockBox 3
21658: 88 unit uTPLb_BlockCipher;                 //LockBox 3
21659: 89 unit uPSI_ValEdit.pas;                  //Delphi VCL
21660: 90 unit uPSI_JvVCLUtils;                   //JCL
21661: 91 unit uPSI_JvDBUtil;                      //JCL
21662: 92 unit uPSI_JvDBUtil;                      //JCL
21663: 93 unit uPSI_JvAppUtils;                   //JCL
21664: 94 unit uPSI_JvCtrlUtils;                  //JCL
21665: 95 unit uPSI_JvFormToHtml;                 //JCL
21666: 96 unit uPSI_JvParsing;                    //JCL
21667: 97 unit uPSI_SerDlg;                        //Toolbox
21668: 98 unit uPSI_Serial;                       //Toolbox
21669: 99 unit uPSI_JvComponent;                 //JCL
21670: 100 unit uPSI_JvCalc;                       //JCL
21671: 101 unit uPSI_JvBdeUtils;                  //JCL
21672: 102 unit uPSI_JvDateUtil;                  //JCL
21673: 103 unit uPSI_JvGenetic;                  //JCL
21674: 104 unit uPSI_JclBase;                     //JCL
21675: 105 unit uPSI_JvUtils;                     //JCL
21676: 106 unit uPSI_JvStringUtil;                //JCL

```

```

21677: 107 unit uPSI_JvStrUtils; //JCL
21678: 108 unit uPSI_JvFileUtil; //JCL
21679: 109 unit uPSI_JvMemoryInfos; //JCL
21680: 110 unit uPSI_JvComputerInfo; //JCL
21681: 111 unit uPSI_JvgCommClasses; //JCL
21682: 112 unit uPSI_JvgLogics; //JCL
21683: 113 unit uPSI_JvLED; //JCL
21684: 114 unit uPSI_JvTurtle; //JCL
21685: 115 unit uPSI_SortThds; unit uPSI_ThSort; //maxbox
21686: 116 unit uPSI_JvgUtils; //JCL
21687: 117 unit uPSI_JvExprParser; //JCL
21688: 118 unit uPSI_HexDump; //Borland
21689: 119 unit uPSI_DBLogDlg; //VCL
21690: 120 unit uPSI_SqlTimSt; //RTL
21691: 121 unit uPSI_JvHtmlParser; //JCL
21692: 122 unit uPSI_JvgXMLSerializer; //JCL
21693: 123 unit uPSI_JvJCLUtils; //JCL
21694: 124 unit uPSI_JvStrings; //JCL
21695: 125 unit uPSI_uTPLb_IntegerUtils; //TurboPower
21696: 126 unit uPSI_uTPLb_HugeCardinal; //TurboPower
21697: 127 unit uPSI_uTPLb_HugeCardinalUtils; //TurboPower
21698: 128 unit uPSI_SynRegExpr; //SynEdit
21699: 129 unit uPSI_StUtils; //SysTools4
21700: 130 unit uPSI_StToHTML; //SysTools4
21701: 131 unit uPSI_StStrms; //SysTools4
21702: 132 unit uPSI_StFIN; //SysTools4
21703: 133 unit uPSI_StAstroP; //SysTools4
21704: 134 unit uPSI_StStat; //SysTools4
21705: 135 unit uPSI_StNetCon; //SysTools4
21706: 136 unit uPSI_StDecMth; //SysTools4
21707: 137 unit uPSI_StOStr; //SysTools4
21708: 138 unit uPSI_StPtrns; //SysTools4
21709: 139 unit uPSI_StNetMsg; //SysTools4
21710: 140 unit uPSI_StMath; //SysTools4
21711: 141 unit uPSI_StExpEng; //SysTools4
21712: 142 unit uPSI_STCRC; //SysTools4
21713: 143 unit uPSI_StExport; //SysTools4
21714: 144 unit uPSI_StExpLog; //SysTools4
21715: 145 unit uPSI_Actnlist; //Delphi VCL
21716: 146 unit uPSI_jpeg; //Borland
21717: 147 unit uPSI_StRandom; //SysTools4
21718: 148 unit uPSI_StDict; //SysTools4
21719: 149 unit uPSI_StBCD; //SysTools4
21720: 150 unit uPSI_StTxtDat; //SysTools4
21721: 151 unit uPSI_StRegEx; //SysTools4
21722: 152 unit uPSI_IMouse; //VCL
21723: 153 unit uPSI_SyncObjs; //VCL
21724: 154 unit uPSI_AsyncCalls; //Hausladen
21725: 155 unit uPSI_ParallelJobs; //Saraiva
21726: 156 unit uPSI_Variants; //VCL
21727: 157 unit uPSI_VarCmplx; //VCL Wolfram
21728: 158 unit uPSI_DTDSchema; //VCL
21729: 159 unit uPSI_ShLwApi; //Brakel
21730: 160 unit uPSI_IBUtils; //VCL
21731: 161 unit uPSI_CheckLst; //VCL
21732: 162 unit uPSI_JvSimpleXml; //JCL
21733: 163 unit uPSI_JclSimpleXml; //JCL
21734: 164 unit uPSI_JvXmlDatabase; //JCL
21735: 165 unit uPSI_JvMaxPixel; //JCL
21736: 166 unit uPSI_JvItemsSearchs; //JCL
21737: 167 unit uPSI_StExpEng2; //SysTools4
21738: 168 unit uPSI_StGenLog; //SysTools4
21739: 169 unit uPSI_JvLogFile; //Jcl
21740: 170 unit uPSI_CPort; //ComPort Lib v4.11
21741: 171 unit uPSI_CPortCtl; //ComPort
21742: 172 unit uPSI_CPortEsc; //ComPort
21743: 173 unit BarCodeScanner; //ComPort
21744: 174 unit uPSI_JvGraph; //JCL
21745: 175 unit uPSI_JvComCtrls; //JCL
21746: 176 unit uPSI_GUITesting; //D Unit
21747: 177 unit uPSI_JvFindFiles; //JCL
21748: 178 unit uPSI_StSystem; //SysTools4
21749: 179 unit uPSI_JvKeyboardStates; //JCL
21750: 180 unit uPSI_JvMail; //JCL
21751: 181 unit uPSI_JclConsole; //JCL
21752: 182 unit uPSI_JclLANman; //JCL
21753: 183 unit uPSI_IdCustomHTTPServer; //Indy
21754: 184 unit IdHTTPServer //Indy
21755: 185 unit uPSI_IdTCPServer; //Indy
21756: 186 unit uPSI_IdSocketHandle; //Indy
21757: 187 unit uPSI_IdIOHandlerSocket; //Indy
21758: 188 unit IdIOHandler; //Indy
21759: 189 unit uPSI_cutils; //Bloodshed
21760: 190 unit uPSI_BoldUtils; //boldsoft
21761: 191 unit uPSI_IdSimpleServer; //Indy
21762: 192 unit uPSI_IdSSLOpenSSL; //Indy
21763: 193 unit uPSI_IdMultipartFormData; //Indy
21764: 194 unit uPSI_SynURIOpener; //SynEdit
21765: 195 unit uPSI_PerlRegEx; //PCRE

```

```

21766: 196 unit uPSI_IdHeaderList;                                //Indy
21767: 197 unit uPSI_StFirst;                                 //SysTools4
21768: 198 unit uPSI_JvCtrls;                                //JCL
21769: 199 unit uPSI_IdTrivialFTPBase;                         //Indy
21770: 200 unit uPSI_IdTrivialFTP;                            //Indy
21771: 201 unit uPSI_IdUDPBase;                             //Indy
21772: 202 unit uPSI_IdUDPClient;                           //Indy
21773: 203 unit uPSI_utypes;                               //for DMath.DLL
21774: 204 unit uPSI_ShellAPI;                            //Borland
21775: 205 unit uPSI_IdRemoteCMDClient;                      //Indy
21776: 206 unit uPSI_IdRemoteCMDServer;                      //Indy
21777: 207 unit IdRexecsServer;                           //Indy
21778: 208 unit IdRexec; (unit uPSI_IdRexec;)             //Indy
21779: 209 unit IdUDPServer;                            //Indy
21780: 210 unit IdTimeUDPServer;                          //Indy
21781: 211 unit IdTimeServer;                            //Indy
21782: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)        //Indy
21783: 213 unit uPSI_IdIPWatch;                           //Indy
21784: 214 unit uPSI_IdIrcServer;                          //Indy
21785: 215 unit uPSI_IdMessageCollection;                  //Indy
21786: 216 unit uPSI_cPEM;                                //Fundamentals 4
21787: 217 unit uPSI_cFundamentUtils;                      //Fundamentals 4
21788: 218 unit uPSI_uwinplot;                           //DMath
21789: 219 unit uPSI_xrtl_util_CPUUtils;                  //ExtentedRTL
21790: 220 unit uPSI_GR32_System;                           //Graphics32
21791: 221 unit uPSI_cFileUtils;                           //Fundamentals 4
21792: 222 unit uPSI_cDateTime; (timemachine)            //Fundamentals 4
21793: 223 unit uPSI_cTimers; (high precision timer)     //Fundamentals 4
21794: 224 unit uPSI_cRandom;                            //Fundamentals 4
21795: 225 unit uPSI_ueval;                             //DMath
21796: 226 unit uPSI_xrtl_net_URIUtils;                 //ExtendedRTL
21797: 227 unit xrtl_net_URIUtils;                      //ExtendedRTL
21798: 228 unit uPSI_ufft; (FFT)                         //DMath
21799: 229 unit uPSI_DBXChannel;                          //Delphi
21800: 230 unit uPSI_DBXIndyChannel;                     //Delphi Indy
21801: 231 unit uPSI_xrtl_util_COMCat;                  //ExtendedRTL
21802: 232 unit uPSI_xrtl_util_StrUtils;                //ExtendedRTL
21803: 233 unit uPSI_xrtl_util_VariantUtils;           //ExtendedRTL
21804: 234 unit uPSI_xrtl_util_FileUtils;              //ExtendedRTL
21805: 235 unit xrtl_util_Compat;                      //ExtendedRTL
21806: 236 unit uPSI_OleAuto;                           //Borland
21807: 237 unit uPSI_xrtl_util_COMUtils;               //ExtendedRTL
21808: 238 unit uPSI_CmAdmCtl;                           //Borland
21809: 239 unit uPSI_ValEdit2;                           //VCL
21810: 240 unit uPSI_GR32; //Graphics32                //Graphics32
21811: 241 unit uPSI_GR32_Image;                         //Graphics32
21812: 242 unit uPSI_xrtl_util_TimeUtils;              //ExtendedRTL
21813: 243 unit uPSI_xrtl_util_TimeZone;               //ExtendedRTL
21814: 244 unit uPSI_xrtl_util_TimeStamp;              //ExtendedRTL
21815: 245 unit uPSI_xrtl_util_Map;                    //ExtendedRTL
21816: 246 unit uPSI_xrtl_util_Set;                   //ExtendedRTL
21817: 247 unit uPSI_CPortMonitor;                      //ComPort
21818: 248 unit uPSI_StIniStm;                           //SysTools4
21819: 249 unit uPSI_GR32_ExtImage;                   //Graphics32
21820: 250 unit uPSI_GR32_OrdinalMaps;                //Graphics32
21821: 251 unit uPSI_GR32_Rasterizers;                //Graphics32
21822: 252 unit uPSI_xrtl_util_Exception;             //ExtendedRTL
21823: 253 unit uPSI_xrtl_util_Value;                 //ExtendedRTL
21824: 254 unit uPSI_xrtl_util_Compare;              //ExtendedRTL
21825: 255 unit uPSI_FlatSB;                           //VCL
21826: 256 unit uPSI_JvAnalogClock;                  //JCL
21827: 257 unit uPSI_JvAlarms;                           //JCL
21828: 258 unit uPSI_JvSQLS;                           //JCL
21829: 259 unit uPSI_JvDBSecur;                         //JCL
21830: 260 unit uPSI_JvDBQBE;                           //JCL
21831: 261 unit uPSI_JvStarfield;                      //JCL
21832: 262 unit uPSI_JVCLMiscal;                      //JCL
21833: 263 unit uPSI_JvProfiler32;                     //JCL
21834: 264 unit uPSI_JvDirectories;                   //JCL
21835: 265 unit uPSI_JclSchedule;                      //JCL
21836: 266 unit uPSI_JclSvcCtrl;                      //JCL
21837: 267 unit uPSI_JvSoundControl;                  //JCL
21838: 268 unit uPSI_JvBDESQLScript;                 //JCL
21839: 269 unit uPSI_JvgDigits;                        //JCL>
21840: 270 unit uPSI_ImgList;                           //TCustomImageList
21841: 271 unit uPSI_JclMIDI;                           //JCL>
21842: 272 unit uPSI_JclWinMidi;                      //JCL>
21843: 273 unit uPSI_JclNTFS;                           //JCL>
21844: 274 unit uPSI_JclAppInst;                      //JCL>
21845: 275 unit uPSI_JvRle;                            //JCL>
21846: 276 unit uPSI_JvRas32;                           //JCL>
21847: 277 unit uPSI_JvImageDrawThread;              //JCL>
21848: 278 unit uPSI_JvImageWindow;                  //JCL>
21849: 279 unit uPSI_JvTransparentForm;              //JCL>
21850: 280 unit uPSI_JvWinDialogs;                   //JCL>
21851: 281 unit uPSI_JvSimLogic;                      //JCL>
21852: 282 unit uPSI_JvSimIndicator;                 //JCL>
21853: 283 unit uPSI_JvSimPID;                       //JCL>
21854: 284 unit uPSI_JvSimPIDLinker,                 //JCL>

```

```

21855: 285 unit uPSI_IdRFCReply; //Indy
21856: 286 unit uPSI_IdIdent; //Indy
21857: 287 unit uPSI_IdIdentServer; //Indy
21858: 288 unit uPSI_JvPatchFile; //JCL
21859: 289 unit uPSI_StNetPfm; //SysTools4
21860: 290 unit uPSI_StNet; //SysTools4
21861: 291 unit uPSI_JclPeImage; //JCL
21862: 292 unit uPSI_JclPrint; //JCL
21863: 293 unit uPSI_JclMime; //JCL
21864: 294 unit uPSI_JvRichEdit; //JCL
21865: 295 unit uPSI_JvDBRichEd; //JCL
21866: 296 unit uPSI_JvDice; //JCL
21867: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
21868: 298 unit uPSI_JvDirFrm; //JCL
21869: 299 unit uPSI_JvDualList; //JCL
21870: 300 unit uPSI_JvSwitch; //JCL
21871: 301 unit uPSI_JvTimerLst; //JCL
21872: 302 unit uPSI_JvMemTable; //JCL
21873: 303 unit uPSI_JvObjStr; //JCL
21874: 304 unit uPSI_StLArr; //SysTools4
21875: 305 unit uPSI_StWmDcpy; //SysTools4
21876: 306 unit uPSI_StText; //SysTools4
21877: 307 unit uPSI_StNTLog; //SysTools4
21878: 308 unit uPSI_xrtl_math_Integer; //ExtendedRTL
21879: 309 unit uPSI_JvImagPrvw; //JCL
21880: 310 unit uPSI_JvFormPatch; //JCL
21881: 311 unit uPSI_JvPicClip; //JCL
21882: 312 unit uPSI_JvDataConv; //JCL
21883: 313 unit uPSI_JvCpuUsage; //JCL
21884: 314 unit uPSI_JclUnitConv_mx2; //JCL
21885: 315 unit JvDualListForm; //JCL
21886: 316 unit uPSI_JvCpuUsage2; //JCL
21887: 317 unit uPSI_JvParserForm; //JCL
21888: 318 unit uPSI_JvJanTreeView; //JCL
21889: 319 unit uPSI_JvTransLED; //JCL
21890: 320 unit uPSI_JvPlaylist; //JCL
21891: 321 unit uPSI_JvFormAutoSize; //JCL
21892: 322 unit uPSI_JvYearGridEditForm; //JCL
21893: 323 unit uPSI_JvMarkupCommon; //JCL
21894: 324 unit uPSI_JvChart; //JCL
21895: 325 unit uPSI_JvXPCore; //JCL
21896: 326 unit uPSI_JvXPCoreUtils; //JCL
21897: 327 unit uPSI_StatsClasses; //mx4
21898: 328 unit uPSI_ExtCtrls2; //VCL
21899: 329 unit uPSI_JvUrlGrabbers; //JCL
21900: 330 unit uPSI_JvXmlTree; //JCL
21901: 331 unit uPSI_JvWavePlayer; //JCL
21902: 332 unit uPSI_JvUnicodeCanvas; //JCL
21903: 333 unit uPSI_JvTFUtils; //JCL
21904: 334 unit uPSI_IdServerIOHandler; //Indy
21905: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
21906: 336 unit uPSI_IdMessageCoder; //Indy
21907: 337 unit uPSI_IdMessageCoderMIME; //Indy
21908: 338 unit uPSI_IdMIMETypes; //Indy
21909: 339 unit uPSI_JvConverter; //JCL
21910: 340 unit uPSI_JvCsvParse; //JCL
21911: 341 unit uPSI_umath; unit uPSI_ugamma; //DMath
21912: 342 unit uPSI_ExcelExport; (Nat:TJsExcelExport) //JCL
21913: 343 unit uPSI_JvDBGridExport; //JCL
21914: 344 unit uPSI_JvgExport; //JCL
21915: 345 unit uPSI_JvSerialMaker; //JCL
21916: 346 unit uPSI_JvWin32; //JCL
21917: 347 unit uPSI_JvPaintFX; //JCL
21918: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
21919: 349 unit uPSI_JvValidators; (preview) //JCL
21920: 350 unit uPSI_JvNTEventLog; //JCL
21921: 351 unit uPSI_ShellZipTool; //mx4
21922: 352 unit uPSI_JvJoystick; //JCL
21923: 353 unit uPSI_JvMailSlots; //JCL
21924: 354 unit uPSI_JclComplex; //JCL
21925: 355 unit uPSI_SynPdf; //Synopse
21926: 356 unit uPSI_Registry; //VCL
21927: 357 unit uPSI_TlHelp32; //VCL
21928: 358 unit uPSI_JclRegistry; //JCL
21929: 359 unit uPSI_JvAirBrush; //JCL
21930: 360 unit uPSI_mORMotReport; //Synopse
21931: 361 unit uPSI_JclLocales; //JCL
21932: 362 unit uPSI_SynEdit; //SynEdit
21933: 363 unit uPSI_SynEditTypes; //SynEdit
21934: 364 unit uPSI_SynMacroRecorder; //SynEdit
21935: 365 unit uPSI_LongIntList; //SynEdit
21936: 366 unit uPSI_devutils; //DevC
21937: 367 unit uPSI_SynEditMiscClasses; //SynEdit
21938: 368 unit uPSI_SynEditRegexSearch; //SynEdit
21939: 369 unit uPSI_SynEditHighlighter; //SynEdit
21940: 370 unit uPSI_SynHighlighterPas; //SynEdit
21941: 371 unit uPSI_JvSearchFiles; //JCL
21942: 372 unit uPSI_SynHighlighterAny; //Lazarus
21943: 373 unit uPSI_SynEditKeyCmds; //SynEdit

```

```

21944: 374 unit uPSI_SynEditMiscProcs; //SynEdit
21945: 375 unit uPSI_SynEditKbdHandler; //SynEdit
21946: 376 unit uPSI_JvAppInst; //JCL
21947: 377 unit uPSI_JvAppEvent; //JCL
21948: 378 unit uPSI_JvAppCommand; //JCL
21949: 379 unit uPSI_JvAnimTitle; //JCL
21950: 380 unit uPSI_JvAnimatedImage; //JCL
21951: 381 unit uPSI_SynEditExport; //SynEdit
21952: 382 unit uPSI_SynExportHTML; //SynEdit
21953: 383 unit uPSI_SynExportRTF; //SynEdit
21954: 384 unit uPSI_SynEditSearch; //SynEdit
21955: 385 unit uPSI_fMain_back //maxbox;
21956: 386 unit uPSI_JvZoom; //JCL
21957: 387 unit uPSI_PMrand; //PM
21958: 388 unit uPSI_JvSticker; //JCL
21959: 389 unit uPSI_XmlVerySimple; //mX4
21960: 390 unit uPSI_Services; //ExtPascal
21961: 391 unit uPSI_ExtPascalUtils; //ExtPascal
21962: 392 unit uPSI_SocketsDelphi; //ExtPascal
21963: 393 unit uPSI_StBarC; //SysTools
21964: 394 unit uPSI_StDbBarC; //SysTools
21965: 395 unit uPSI_StBarPN; //SysTools
21966: 396 unit uPSI_StDbPNBC; //SysTools
21967: 397 unit uPSI_StDb2DBC; //SysTools
21968: 398 unit uPSI_StMoney; //SysTools
21969: 399 unit uPSI_JvForth; //JCL
21970: 400 unit uPSI_RestRequest; //mX4
21971: 401 unit uPSI_HttpRESTConnectionIndy; //mX4
21972: 402 unit uPSI_JvXmlDatabase; //update //JCL
21973: 403 unit uPSI_StAstro; //SysTools
21974: 404 unit uPSI_StSort; //SysTools
21975: 405 unit uPSI_StDate; //SysTools
21976: 406 unit uPSI_StDateSt; //SysTools
21977: 407 unit uPSI_StBase; //SysTools
21978: 408 unit uPSI_StVInfo; //SysTools
21979: 409 unit uPSI_JvBrowseFolder; //JCL
21980: 410 unit uPSI_JvBoxProcs; //JCL
21981: 411 unit uPSI_urandom; (unit uranuvag;) //DMath
21982: 412 unit uPSI_usimann; (unit ugenalg;) //DMath
21983: 413 unit uPSI_JvHighlighter; //JCL
21984: 414 unit uPSI_Diff; //mX4
21985: 415 unit uPSI_SpringWinAPI; //DSpring
21986: 416 unit uPSI_StBits; //SysTools
21987: 417 unit uPSI_TomDBQue; //mX4
21988: 418 unit uPSI_MultilangTranslator; //mX4
21989: 419 unit uPSI_HyperLabel; //mX4
21990: 420 unit uPSI_Starter; //mX4
21991: 421 unit uPSI_FileAssocs; //devC
21992: 422 unit uPSI_devFileMonitorX; //devC
21993: 423 unit uPSI_devrun; //devC
21994: 424 unit uPSI_devExec; //devC
21995: 425 unit uPSI_oyzUtils; //devC
21996: 426 unit uPSI_DosCommand; //devC
21997: 427 unit uPSI_CppTokenizer; //devC
21998: 428 unit uPSI_JvHLPParser; //devC
21999: 429 unit uPSI_JclMapi; //JCL
22000: 430 unit uPSI_JclShell; //JCL
22001: 431 unit uPSI_JclCOM; //JCL
22002: 432 unit uPSI_GR32_Math; //Graphics32
22003: 433 unit uPSI_GR32_LowLevel; //Graphics32
22004: 434 unit uPSI_SimpleHl; //mX4
22005: 435 unit uPSI_GR32_Filters; //Graphics32
22006: 436 unit uPSI_GR32_VectorMaps; //Graphics32
22007: 437 unit uPSI_cXMLFunctions; //Fundamentals 4
22008: 438 unit uPSI_JvTimer; //JCL
22009: 439 unit uPSI_cHTTPUtils; //Fundamentals 4
22010: 440 unit uPSI_cTLSUtils; //Fundamentals 4
22011: 441 unit uPSI_JclGraphics; //JCL
22012: 442 unit uPSI_JclSynch; //JCL
22013: 443 unit uPSI_IdTelnet; //Indy
22014: 444 unit uPSI_IdTelnetServer; //Indy
22015: 445 unit uPSI_IdEcho; //Indy
22016: 446 unit uPSI_IdEchoServer; //Indy
22017: 447 unit uPSI_IdEchoUDP; //Indy
22018: 448 unit uPSI_IdEchoUDPServer; //Indy
22019: 449 unit uPSI_IdSocks; //Indy
22020: 450 unit uPSI_IdAntiFreezeBase; //Indy
22021: 451 unit uPSI_IdHostnameServer; //Indy
22022: 452 unit uPSI_IdTunnelCommon; //Indy
22023: 453 unit uPSI_IdTunnelMaster; //Indy
22024: 454 unit uPSI_IdTunnelSlave; //Indy
22025: 455 unit uPSI_IdRSH; //Indy
22026: 456 unit uPSI_IdRSHServer; //Indy
22027: 457 unit uPSI_Spring_Cryptography_Utils; //Spring4Delphi
22028: 458 unit uPSI_MapReader; //devC
22029: 459 unit uPSI_LibTar; //devC
22030: 460 unit uPSI_IdStack; //Indy
22031: 461 unit uPSI_IdBlockCipherIntercept; //Indy
22032: 462 unit uPSI_IdChargenServer; //Indy

```

```

22033: 463 unit uPSI_IdFTPServer; //Indy
22034: 464 unit uPSI_IdException; //Indy
22035: 465 unit uPSI_utexplot; //DMath
22036: 466 unit uPSI_uwinstr; //DMath
22037: 467 unit uPSI_VarRecUtils; //devC
22038: 468 unit uPSI_JvStringListToHtml; //JCL
22039: 469 unit uPSI_JvStringHolder; //JCL
22040: 470 unit uPSI_IdCoder; //Indy
22041: 471 unit uPSI_SynHighlighterDfm; //Synedit
22042: 472 unit uHighlighterProcs; in 471 //Synedit
22043: 473 unit uPSI_LazFileUtils; //LCL
22044: 474 unit uPSI_IDECmdLine; //LCL
22045: 475 unit uPSI_lazMasks; //LCL
22046: 476 unit uPSI_ip_misci; //mX4
22047: 477 unit uPSI_Barcodes; //LCL
22048: 478 unit uPSI_SimpleXML; //LCL
22049: 479 unit uPSI_JclIniFiles; //JCL
22050: 480 unit uPSI_D2XXUnit; {$X-} //FTDI
22051: 481 unit uPSI_JclDateTime; //JCL
22052: 482 unit uPSI_JclEDI; //JCL
22053: 483 unit uPSI_JclMiscel2; //JCL
22054: 484 unit uPSI_JclValidation; //JCL
22055: 485 unit uPSI_JclAnsiStrings; {-PString} //JCL
22056: 486 unit uPSI_SynEditMiscProcs2; //Synedit
22057: 487 unit uPSI_JclStreams; //JCL
22058: 488 unit uPSI_QRCode; //mX4
22059: 489 unit uPSI_BlockSocket; //ExtPascal
22060: 490 unit uPSI_Masks_Utils; //VCL
22061: 491 unit uPSI_synautil; //Synapse!
22062: 492 unit uPSI_JclMath_Class; //JCL RTL
22063: 493 unit ugamdist; //Gamma function //DMath
22064: 494 unit uibeta, ucorrel; //IBeta //DMath
22065: 495 unit uPSI_SRMgr; //mX4
22066: 496 unit uPSI_HotLog; //mX4
22067: 497 unit uPSI_DebugBox; //mX4
22068: 498 unit uPSI_ustrings; //DMath
22069: 499 unit uPSI_uregtest; //DMath
22070: 500 unit uPSI_usimplex; //DMath
22071: 501 unit uPSI_uhyper; //DMath
22072: 502 unit uPSI_IdHL7; //Indy
22073: 503 unit uPSI_IdIPMCastBase; //Indy
22074: 504 unit uPSI_IdIPMCastServer; //Indy
22075: 505 unit uPSI_IdIPMCastClient; //Indy
22076: 506 unit uPSI_unlfit; //nlregression //DMath
22077: 507 unit uPSI_IdRawHeaders; //Indy
22078: 508 unit uPSI_IdRawClient; //Indy
22079: 509 unit uPSI_IdRawFunctions; //Indy
22080: 510 unit uPSI_IdTCPStream; //Indy
22081: 511 unit uPSI_IdSNPP; //Indy
22082: 512 unit uPSI_St2DBarC; //SysTools
22083: 513 unit uPSI_ImageWin; //FTL //VCL
22084: 514 unit uPSI_CustomDrawTreeView; //FTL //VCL
22085: 515 unit uPSI_GraphWin; //FTL //VCL
22086: 516 unit uPSI_actionMain; //FTL //VCL
22087: 517 unit uPSI_StSpawn; //SysTools
22088: 518 unit uPSI_CtlPanel; //VCL
22089: 519 unit uPSI_IdLPR; //Indy
22090: 520 unit uPSI_SockRequestInterpreter; //Indy
22091: 521 unit uPSI_ulambert; //DMath
22092: 522 unit uPSI_ucholesk; //DMath
22093: 523 unit uPSI_SimpleDS; //VCL
22094: 524 unit uPSI_DBXSqlScanner; //VCL
22095: 525 unit uPSI_DBXMetaDataTable; //VCL
22096: 526 unit uPSI_Chart; //TEE
22097: 527 unit uPSI_TeeProcs; //TEE
22098: 528 unit mXBDEUtils; //mX4
22099: 529 unit uPSI_MDIEdit; //VCL
22100: 530 unit uPSI_CopyPrsr; //VCL
22101: 531 unit uPSI_SockApp; //VCL
22102: 532 unit uPSI_AppEvnts; //VCL
22103: 533 unit uPSI_ExtActns; //VCL
22104: 534 unit uPSI_TeEngine; //TEE
22105: 535 unit uPSI_CoolMain; //browser //VCL
22106: 536 unit uPSI_StCRC; //SysTools
22107: 537 unit uPSI_StDecMth2; //SysTools
22108: 538 unit uPSI_frmExportMain; //Synedit
22109: 539 unit uPSI_SynDBEdit; //Synedit
22110: 540 unit uPSI_SynEditWildcardSearch; //Synedit
22111: 541 unit uPSI_BoldComUtils; //BOLD
22112: 542 unit uPSI_BoldIsoDateTime; //BOLD
22113: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
22114: 544 unit uPSI_BoldXMLRequests; //BOLD
22115: 545 unit uPSI_BoldStringList; //BOLD
22116: 546 unit uPSI_BoldFileHandler; //BOLD
22117: 547 unit uPSI_BoldContainers; //BOLD
22118: 548 unit uPSI_BoldQueryUserDlg; //BOLD
22119: 549 unit uPSI_BoldWinINet; //BOLD
22120: 550 unit uPSI_BoldQueue; //BOLD
22121: 551 unit uPSI_JvPcx; //JCL

```

```

22122: 552 unit uPSI_IdWhois; //Indy
22123: 553 unit uPSI_IdWhoIsServer; //Indy
22124: 554 unit uPSI_IdGopher; //Indy
22125: 555 unit uPSI_IdDateTimeStamp; //Indy
22126: 556 unit uPSI_IdDayTimeServer; //Indy
22127: 557 unit uPSI_IdDayTimeUDP; //Indy
22128: 558 unit uPSI_IdDayTimeUDPServer; //Indy
22129: 559 unit uPSI_IdDICTServer; //Indy
22130: 560 unit uPSI_IdDiscardServer; //Indy
22131: 561 unit uPSI_IdDiscardUDPServer; //Indy
22132: 562 unit uPSI_IdMappedFTP; //Indy
22133: 563 unit uPSI_IdMappedPortTCP; //Indy
22134: 564 unit uPSI_IdGopherServer; //Indy
22135: 565 unit uPSI_IdQotdServer; //Indy
22136: 566 unit uPSI_JvRgbToHtml; //JCL
22137: 567 unit uPSI_JvRemLog; //JCL
22138: 568 unit uPSI_JvSysComp; //JCL
22139: 569 unit uPSI_JvTMTL; //JCL
22140: 570 unit uPSI_JvWinampAPI; //JCL
22141: 571 unit uPSI_MSysUtils; //mX4
22142: 572 unit uPSI_ESBMaths; //ESB
22143: 573 unit uPSI_ESBMaths2; //ESB
22144: 574 unit uPSI_uLkJSON; //Lk
22145: 575 unit uPSI_ZURL; //Zeos
22146: 576 unit uPSI_ZSysUtils; //Zeos
22147: 577 unit unaUtils internals //UNA
22148: 578 unit uPSI_ZMatchPattern; //Zeos
22149: 579 unit uPSI_ZClasses; //Zeos
22150: 580 unit uPSI_ZCollections; //Zeos
22151: 581 unit uPSI_ZEncoding; //Zeos
22152: 582 unit uPSI_IdRawBase; //Indy
22153: 583 unit uPSI_IdNTLM; //Indy
22154: 584 unit uPSI_IdNNTP; //Indy
22155: 585 unit uPSI_usniffer; //PortScanForm //mX4
22156: 586 unit uPSI_IdCoderMIME; //Indy
22157: 587 unit uPSI_IdCoderUUE; //Indy
22158: 588 unit uPSI_IdCoderXXE; //Indy
22159: 589 unit uPSI_IdCoder3to4; //Indy
22160: 590 unit uPSI_IdCookie; //Indy
22161: 591 unit uPSI_IdCookieManager; //Indy
22162: 592 unit uPSI_WDOSocketUtils; //WDOS
22163: 593 unit uPSI_WDOSPlcUtils; //WDOS
22164: 594 unit uPSI_WDOSPorts; //WDOS
22165: 595 unit uPSI_WDOSResolvers; //WDOS
22166: 596 unit uPSI_WDOSTimers; //WDOS
22167: 597 unit uPSI_WDOSPlcs; //WDOS
22168: 598 unit uPSI_WDOSPneumatics; //WDOS
22169: 599 unit uPSI_IdFingerServer; //Indy
22170: 600 unit uPSI_IdDNSResolver; //Indy
22171: 601 unit uPSI_IdHTTPWebBrokerBridge; //Indy
22172: 602 unit uPSI_IdIntercept; //Indy
22173: 603 unit uPSI_IdIPMCastBase; //Indy
22174: 604 unit uPSI_IdLogBase; //Indy
22175: 605 unit uPSI_IdIOHandlerStream; //Indy
22176: 606 unit uPSI_IdMappedPortUDP; //Indy
22177: 607 unit uPSI_IdQOTDUDPServer; //Indy
22178: 608 unit uPSI_IdQOTDUDP; //Indy
22179: 609 unit uPSI_IdSysLog; //Indy
22180: 610 unit uPSI_IdSysLogServer; //Indy
22181: 611 unit uPSI_IdSysLogMessage; //Indy
22182: 612 unit uPSI_IdTimeServer; //Indy
22183: 613 unit uPSI_IdTimeUDP; //Indy
22184: 614 unit uPSI_IdTimeUDPServer; //Indy
22185: 615 unit uPSI_IdUserAccounts; //Indy
22186: 616 unit uPSI_TextUtils; //mX4
22187: 617 unit uPSI_MandelbrotEngine; //mX4
22188: 618 unit uPSI_delphi_arduino_Unit1; //mX4
22189: 619 unit uPSI_DTDSchema2; //mX4
22190: 620 unit uPSI_fplotMain; //DMath
22191: 621 unit uPSI_FindfileIter; //mX4
22192: 622 unit uPSI_PppState; (JclStrHashMap) //PPP
22193: 623 unit uPSI_PppParser; //PPP
22194: 624 unit uPSI_PppLexer; //PPP
22195: 625 unit uPSI_PCharUtils; //PPP
22196: 626 unit uPSI_uJSON; //WU
22197: 627 unit uPSI_JclStrHashMap; //JCL
22198: 628 unit uPSI_JclHookExcept; //JCL
22199: 629 unit uPSI_EncdDecd; //VCL
22200: 630 unit uPSI_SockAppReg; //VCL
22201: 631 unit uPSI_PJFileHandle; //PJ
22202: 632 unit uPSI_PJEnvVars; //PJ
22203: 633 unit uPSI_PJPipe; //PJ
22204: 634 unit uPSI_PJPipeFilters; //PJ
22205: 635 unit uPSI_PJConsoleApp; //PJ
22206: 636 unit uPSI_UConsoleAppEx; //PJ
22207: 637 unit uPSI_DbxSocketChannelNative; //VCL
22208: 638 unit uPSI_DbxDataGenerator; //VCL
22209: 639 unit uPSI_DBXClient; //VCL
22210: 640 unit uPSI_IdLogEvent; //Indy

```

```

22211: 641 unit uPSI_Reversi;                                //mX4
22212: 642 unit uPSI_Geometry;                             //mX4
22213: 643 unit uPSI_IdSMTPServer;                          //Indy
22214: 644 unit uPSI_Textures;                             //mX4
22215: 645 unit uPSI_IBX;                                 //VCL
22216: 646 unit uPSI_IWDBCommon;                           //VCL
22217: 647 unit uPSI_SortGrid;                            //mX4
22218: 648 unit uPSI_IB;                                 //VCL
22219: 649 unit uPSI_IBScript;                            //VCL
22220: 650 unit uPSI_JvCSVBaseControls;                  //JCL
22221: 651 unit uPSI_Jvg3DColors;                         //JCL
22222: 652 unit uPSI_JvHLEditor; //beat                  //JCL
22223: 653 unit uPSI_JvShellHook;                          //JCL
22224: 654 unit uPSI_DBCommon2;                           //VCL
22225: 655 unit uPSI_JvSHFileOperation;                  //JCL
22226: 656 unit uPSI_uFileExport;                          //mX4
22227: 657 unit uPSI_JvDialogs;                           //JCL
22228: 658 unit uPSI_JvDBTreeView;                         //JCL
22229: 659 unit uPSI_JvDBUltimGrid;                      //JCL
22230: 660 unit uPSI_JvDBQueryParamsForm;                //JCL
22231: 661 unit uPSI_JvExControls;                        //JCL
22232: 662 unit uPSI_JvBDEMemTable;                      //JCL
22233: 663 unit uPSI_JvCommStatus;                        //JCL
22234: 664 unit uPSI_JvMailSlots2;                        //JCL
22235: 665 unit uPSI_JvgWinMask;                          //JCL
22236: 666 unit uPSI_StEclipse;                           //SysTools
22237: 667 unit uPSI_StMime;                             //SysTools
22238: 668 unit uPSI_StList;                            //SysTools
22239: 669 unit uPSI_StMerge;                            //SysTools
22240: 670 unit uPSI_StStrS;                            //SysTools
22241: 671 unit uPSI_StTree;                            //SysTools
22242: 672 unit uPSI_StVArr;                            //SysTools
22243: 673 unit uPSI_StRegInи;                           //SysTools
22244: 674 unit uPSI_urkf;                             //DMath
22245: 675 unit uPSI_usvd;                            //DMath
22246: 676 unit uPSI_DepWalkUtils;                     //JCL
22247: 677 unit uPSI_OptionsFrm;                        //JCL
22248: 678 unit yuvconverts;                           //mX4
22249: 679 uPSI_JvPropAutoSave;                         //JCL
22250: 680 uPSI_AclAPI;                             //alcinoe
22251: 681 uPSI_AviCap;                            //alcinoe
22252: 682 uPSI_ALAVLBinaryTree;                      //alcinoe
22253: 683 uPSI_ALFcнMisc;                           //alcinoe
22254: 684 uPSI_ALStringList;                         //alcinoe
22255: 685 uPSI_ALQuickSortList;                      //alcinoe
22256: 686 uPSI_ALStaticText;                         //alcinoe
22257: 687 uPSI_ALJSONDoc;                           //alcinoe
22258: 688 uPSI_ALGSMComm;                           //alcinoe
22259: 689 uPSI_ALWindows;                           //alcinoe
22260: 690 uPSI_ALMultiPartFormDataParser;           //alcinoe
22261: 691 uPSI_ALHttpCommon;                         //alcinoe
22262: 692 uPSI_ALWebSpider;                          //alcinoe
22263: 693 uPSI_ALHttpClient;                         //alcinoe
22264: 694 uPSI_ALFcнHTML;                           //alcinoe
22265: 695 uPSI_ALFTPClient;                          //alcinoe
22266: 696 uPSI_ALInternetMessageCommon;            //alcinoe
22267: 697 uPSI_ALWininetHttpClient;                 //alcinoe
22268: 698 uPSI_ALWininetFTPClient;                  //alcinoe
22269: 699 uPSI_ALWinHttpWrapper;                    //alcinoe
22270: 700 uPSI_ALWinHttpclient;                      //alcinoe
22271: 701 uPSI_ALFcнWinSock;                         //alcinoe
22272: 702 uPSI_ALFcнSQL;                            //alcinoe
22273: 703 uPSI_ALFcнCGI;                           //alcinoe
22274: 704 uPSI_ALFcнExecute;                         //alcinoe
22275: 705 uPSI_ALFcнFile;                           //alcinoe
22276: 706 uPSI_ALFcнMime;                           //alcinoe
22277: 707 uPSI_ALPhpRunner;                          //alcinoe
22278: 708 uPSI_ALGraphic;                           //alcinoe
22279: 709 uPSI_ALIniFiles;                           //alcinoe
22280: 710 uPSI_ALMemCachedClient;                   //alcinoe
22281: 711 unit uPSI_MyGrids;                          //mX4
22282: 712 uPSI_ALMultiPartMixedParser;              //alcinoe
22283: 713 uPSI_ALSMTPCClient;                       //alcinoe
22284: 714 uPSI_ALNNNTPCClient;                     //alcinoe
22285: 715 uPSI_ALHintBalloon;                       //alcinoe
22286: 716 unit uPSI_ALXmlDoc;                        //alcinoe
22287: 717 unit uPSI_IPCThrd;                         //VCL
22288: 718 unit uPSI_MonForm;                          //VCL
22289: 719 unit uPSI_TeCanvas;                         //Orpheus
22290: 720 unit uPSI_Ovcmisc;                          //Orpheus
22291: 721 unit uPSI_ovcfiler;                        //Orpheus
22292: 722 unit uPSI_ovcstate;                        //Orpheus
22293: 723 unit uPSI_ovccoco;                         //Orpheus
22294: 724 unit uPSI_ovcrvexp;                        //Orpheus
22295: 725 unit uPSI_OvcFormatSettings;              //Orpheus
22296: 726 unit uPSI_OvcUtils;                         //Orpheus
22297: 727 unit uPSI_ovcstore;                        //Orpheus
22298: 728 unit uPSI_ovcstr;                           //Orpheus
22299: 729 unit uPSI_ovcmru;                          //Orpheus

```

```

22300: 730 unit uPSI_ovccmd;                                //Orpheus
22301: 731 unit uPSI_ovctimer;                             //Orpheus
22302: 732 unit uPSI_ovcintl;                             //Orpheus
22303: 733 uPSI_AfCircularBuffer;                         //AsyncFree
22304: 734 uPSI_AfUtils;                                 //AsyncFree
22305: 735 uPSI_AfSafeSync;                             //AsyncFree
22306: 736 uPSI_AfComPortCore;                           //AsyncFree
22307: 737 uPSI_AfComPort;                             //AsyncFree
22308: 738 uPSI_AfPortControls;                          //AsyncFree
22309: 739 uPSI_AfDataDispatcher;                        //AsyncFree
22310: 740 uPSI_AfViewers;                            //AsyncFree
22311: 741 uPSI_AfDataTerminal;                          //AsyncFree
22312: 742 uPSI_SimplePortMain;                         //AsyncFree
22313: 743 unit uPSI_ovcclock;                           //Orpheus
22314: 744 unit uPSI_o32intlst;                          //Orpheus
22315: 745 unit uPSI_o32ledlabel;                        //Orpheus
22316: 746 unit uPSI_AlMySQLClient;                      //alcinoe
22317: 747 unit uPSI_ALFBXClient;                        //alcinoe
22318: 748 unit uPSI_ALFcnsSQL;                          //alcinoe
22319: 749 unit uPSI_AsyncTimer;                         //mX4
22320: 750 unit uPSI_ApplicationFileIO;                  //mX4
22321: 751 unit uPSI_PsAPI;                            //VCLé
22322: 752 uPSI_ovcuser;                               //Orpheus
22323: 753 uPSI_ovcurl;                                //Orpheus
22324: 754 uPSI_ovcvlb;                                //Orpheus
22325: 755 uPSI_ovccolor;                             //Orpheus
22326: 756 uPSI_ALFBXLib;                            //alcinoe
22327: 757 uPSI_ovcmeter;                            //Orpheus
22328: 758 uPSI_ovcpeakm;                            //Orpheus
22329: 759 uPSI_O32BGSty;                            //Orpheus
22330: 760 uPSI_ovcbidi;                            //Orpheus
22331: 761 uPSI_ovctcarry;                           //Orpheus
22332: 762 uPSI_DXPUtils;                            //mX4
22333: 763 uPSI_ALMultiPartBaseParser;                 //alcinoe
22334: 764 uPSI_ALMultiPartAlternativeParser;          //alcinoe
22335: 765 uPSI_ALPOP3Client;                         //alcinoe
22336: 766 uPSI_SmallUtils;                           //mX4
22337: 767 uPSI_MakeApp;                            //mX4
22338: 768 uPSI_O32MouseMon;                          //Orpheus
22339: 769 uPSI_OvcCache;                            //Orpheus
22340: 770 uPSI_ovccalc;                            //Orpheus
22341: 771 uPSI_Joystick;                           //OpenGL
22342: 772 uPSI_ScreenSaver;                          //OpenGL
22343: 773 uPSI_XCollection;                         //OpenGL
22344: 774 uPSI_Polynomials;                         //OpenGL
22345: 775 uPSI_PersistentClasses, //9.86           //OpenGL
22346: 776 uPSI_VectorLists;                          //OpenGL
22347: 777 uPSI_XOpenGL;                            //OpenGL
22348: 778 uPSI_MeshUtils;                           //OpenGL
22349: 779 unit uPSI_JclSysUtils;                    //JCL
22350: 780 unit uPSI_JclBorlandTools;                 //JCL
22351: 781 unit JclFileUtils_max;                     //JCL
22352: 782 uPSI_AfDataControls;                      //AsyncFree
22353: 783 uPSI_GLSilhouette;                        //OpenGL
22354: 784 uPSI_JclSysUtils_class;                   //JCL
22355: 785 uPSI_JclFileUtils_class;                  //JCL
22356: 786 uPSI_FileUtil;                            //JCL
22357: 787 uPSI_changefind;                           //mX4
22358: 788 uPSI_cmdIntf;                            //mX4
22359: 789 uPSI_fservice;                            //mX4
22360: 790 uPSI_Keyboard;                            //OpenGL
22361: 791 uPSI_VRMLParser;                          //OpenGL
22362: 792 uPSI_GLFileVRML;                         //OpenGL
22363: 793 uPSI_Octree;                            //OpenGL
22364: 794 uPSI_GLPolyhedron;                        //OpenGL
22365: 795 uPSI_GLCrossPlatform;                     //OpenGL
22366: 796 uPSI_GLParticles;                         //OpenGL
22367: 797 uPSI_GLNavigator;                         //OpenGL
22368: 798 uPSI_GLStarRecord;                        //OpenGL
22369: 799 uPSI_GLTextureCombiners;                 //OpenGL
22370: 800 uPSI_GLCanvas;                            //OpenGL
22371: 801 uPSI_GeometryBB;                          //OpenGL
22372: 802 uPSI_GeometryCoordinates;                 //OpenGL
22373: 803 uPSI_VectorGeometry;                      //OpenGL
22374: 804 uPSI_BumpMapping;                         //OpenGL
22375: 805 uPSI_TGA;                                //OpenGL
22376: 806 uPSI_GLVectorFileObjects;                 //OpenGL
22377: 807 uPSI_IMM;                                //VCL
22378: 808 uPSI_CategoryButtons;                     //VCL
22379: 809 uPSI_ButtonGroup;                         //VCL
22380: 810 uPSI_DbExcept;                            //VCL
22381: 811 uPSI_AxCtrls;                            //VCL
22382: 812 uPSI_GL_actorUnit1;                      //OpenGL
22383: 813 uPSI_StdVCL;                            //VCL
22384: 814 unit CurvesAndSurfaces;                  //OpenGL
22385: 815 uPSI_DataAwareMain;                       //AsyncFree
22386: 816 uPSI_TabNotBk;                            //VCL
22387: 817 uPSI_udwsfiler;                           //mX4
22388: 818 uPSI_synaip;                            //Synapse!

```

```

22389: 819 uPSI_synacode; //Synapse
22390: 820 uPSI_synachar; //Synapse
22391: 821 uPSI_synamisc; //Synapse
22392: 822 uPSI_synaser; //Synapse
22393: 823 uPSI_synaicnv; //Synapse
22394: 824 uPSI_tlntsend; //Synapse
22395: 825 uPSI_pingsend; //Synapse
22396: 826 uPSI_blksock; //Synapse
22397: 827 uPSI_asnlutil; //Synapse
22398: 828 uPSI_dnssend; //Synapse
22399: 829 uPSI_clamsend; //Synapse
22400: 830 uPSI_ldapsend; //Synapse
22401: 831 uPSI_mimemess; //Synapse
22402: 832 uPSI_slogsend; //Synapse
22403: 833 uPSI_mimepart; //Synapse
22404: 834 uPSI_mimeinln; //Synapse
22405: 835 uPSI_ftpsend; //Synapse
22406: 836 uPSI_ftptsend; //Synapse
22407: 837 uPSI_httpsend; //Synapse
22408: 838 uPSI_sntpsend; //Synapse
22409: 839 uPSI_smtpsend; //Synapse
22410: 840 uPSI_snmpsend; //Synapse
22411: 841 uPSI_imapsend; //Synapse
22412: 842 uPSI_pop3send; //Synapse
22413: 843 uPSI_ntpsend; //Synapse
22414: 844 uPSI_ssl_cryptlib; //Synapse
22415: 845 uPSI_ssl_openssl; //Synapse
22416: 846 uPSI_synthtpp_daemon; //Synapse
22417: 847 uPSI_NetWork; //mX4
22418: 848 uPSI_PingThread; //Synapse
22419: 849 uPSI_JvThreadTimer; //JCL
22420: 850 unit uPSI_wwSystem; //InfoPower
22421: 851 unit uPSI_IdComponent; //Indy
22422: 852 unit uPSI_IdIOHandlerThrottle; //Indy
22423: 853 unit uPSI_Themes; //VCL
22424: 854 unit uPSI_StdStyleActnCtrls; //VCL
22425: 855 unit uPSI_UDDIHelper; //VCL
22426: 856 unit uPSI_IdIMAP4Server; //Indy
22427: 857 uPSI_VariantSymbolTable; //VCL //3.9.9.92
22428: 858 uPSI_udf_glob; //mX4
22429: 859 uPSI_TabGrid; //VCL
22430: 860 uPSI_JsDBTreeView; //mX4
22431: 861 uPSI_JsSendMail; //mX4
22432: 862 uPSI_dbTvRecordList; //mX4
22433: 863 uPSI_TreeWvEx; //mX4
22434: 864 uPSI_ECDataLink; //mX4
22435: 865 uPSI_dbTree; //mX4
22436: 866 uPSI_dbTreeCBox; //mX4
22437: 867 unit uPSI_Debug; //TfrmDebug //mX4
22438: 868 uPSI_TimeFncs; //mX4
22439: 869 uPSI_FileIntf; //VCL
22440: 870 uPSI_SockTransport; //RTL
22441: 871 unit uPSI_WinInet; //RTL
22442: 872 unit uPSI_WWSTR; //mX4
22443: 873 uPSI_DBLookup; //VCL
22444: 874 uPSI_Hotspot; //mX4
22445: 875 uPSI_HList; //History List //mX4
22446: 876 unit uPSI_DrTable; //VCL
22447: 877 uPSI_TConnect; //VCL
22448: 878 uPSI_DataBkr; //VCL
22449: 879 uPSI_HTTPIntr; //VCL
22450: 880 unit uPSI_Mathbox; //mX4
22451: 881 uPSI_cyIndy; //cY
22452: 882 uPSI_cySysUtils; //cY
22453: 883 uPSI_cyWinUtils; //cY
22454: 884 uPSI_cyStrUtils; //cY
22455: 885 uPSI_cyObjUtils; //cY
22456: 886 uPSI_cyDateUtils; //cY
22457: 887 uPSI_cyBDE; //cY
22458: 888 uPSI_cyClasses; //cY
22459: 889 uPSI_cyGraphics; //3.9.9.94_2 //cY
22460: 890 unit uPSI_cyTypes; //cY
22461: 891 uPSI_JvDateTimePicker; //JCL
22462: 892 uPSI_JvCreateProcess; //JCL
22463: 893 uPSI_JvEasterEgg; //JCL
22464: 894 uPSI_WinSvc; //3.9.9.94_3 //VCL
22465: 895 uPSI_SvcMgr //VCL
22466: 896 unit uPSI_JvPickDate; //JCL
22467: 897 unit uPSI_JvNotify; //JCL
22468: 898 uPSI_JvStrHlder //JCL
22469: 899 unit uPSI_JclNTFS2; //JCL
22470: 900 uPSI_Jcl8087 //math coprocessor //JCL
22471: 901 uPSI_JvAddPrinter //JCL
22472: 902 uPSI_JvCabFile //JCL
22473: 903 uPSI_JvDataEmbedded; //JCL
22474: 904 unit uPSI_U_HexView; //mX4
22475: 905 uPSI_UWavein4; //mX4
22476: 906 uPSI_AMixer; //mX4
22477: 907 uPSI_JvaScrollText; //mX4

```

```

22478: 908 uPSI_JvArrow; //mX4
22479: 909 unit uPSI_UrlMon; //mX4
22480: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas' //mX4
22481: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain; //DFF
22482: 912 unit uPSI_DFFUtils; //DFF
22483: 913 unit uPSI_MathsLib; //DFF
22484: 914 uPSI_UIntList; //DFF
22485: 915 uPSI_UGetParens; //DFF
22486: 916 unit uPSI_UGeometry; //DFF
22487: 917 unit uPSI_UAstronomy; //DFF
22488: 918 unit uPSI_UCardComponentV2; //DFF
22489: 919 unit uPSI_UTGraphSearch; //DFF
22490: 920 unit uPSI_UParser10; //DFF
22491: 921 unit uPSI_cyIEUtils; //cY
22492: 922 unit uPSI_UcomboV2; //DFF
22493: 923 uPSI_cyBaseComm; //cY
22494: 924 uPSI_cyAppInstances; //cY
22495: 925 uPSI_cyAttract; //cY
22496: 926 uPSI_cyDERUtils //cY
22497: 927 unit uPSI_cyDocER; //cY
22498: 928 unit uPSI_ODBC; //mX
22499: 929 unit uPSI_AssocExec; //mX
22500: 930 uPSI_cyBaseCommRoomConnector; //cY
22501: 931 uPSI_cyCommRoomConnector; //cY
22502: 932 uPSI_cyCommunicate; //cY
22503: 933 uPSI_cyImage; //cY
22504: 934 uPSI_cyBaseContainer; //cY
22505: 935 uPSI_cyModalContainer; //cY
22506: 936 uPSI_cyFlyingContainer; //cY
22507: 937 uPSI_RegStr; //VCL
22508: 938 uPSI_HtmlHelpViewer; //VCL
22509: 939 unit uPSI_cyIniform; //cY
22510: 940 unit uPSI_cyVirtualGrid; //cY
22511: 941 uPSI_Profiler; //DA
22512: 942 uPSI_BackgroundWorker; //DA
22513: 943 uPSI_WavePlay; //DA
22514: 944 uPSI_WaveTimer; //DA
22515: 945 uPSI_WaveUtils; //DA
22516: 946 uPSI_NamedPipes; //TB
22517: 947 uPSI_NamedPipeServer; //TB
22518: 948 unit uPSI_process; //TB
22519: 949 unit uPSI_DPUtils; //TB
22520: 950 unit uPSI_CommonTools; //TB
22521: 951 uPSI_DataSendToWeb; //TB
22522: 952 uPSI_StarCalc; //TB
22523: 953 uPSI_D2_XPVistaHelperU //TB
22524: 954 unit uPSI_NetTools; //TB
22525: 955 unit uPSI_Pipes; //TB
22526: 956 uPSI_ProcessUnit; //mX
22527: 957 uPSI_adGSM; //mX
22528: 958 unit uPSI_BetterADODataSet; //mX
22529: 959 unit uPSI_AdSelCom; //FTT //mX
22530: 960 unit unit uPSI_dwsXPlatform; //DWS
22531: 961 uPSI_AdSocket; //mX Turbo Power
22532: 962 uPSI_AdPacket; //mX
22533: 963 uPSI_AdPort; //mX
22534: 964 uPSI_PathFunc; //Inno
22535: 965 uPSI_CmnFunc; //Inno
22536: 966 uPSI_CmnFunc2; //Inno Setup //Inno
22537: 967 unit uPSI_BitmapImage; //mX4
22538: 968 unit uPSI_ImageGrabber; //mX4
22539: 969 uPSI_SecurityFunc; //Inno
22540: 970 uPSI_RedirFunc; //Inno
22541: 971 uPSI_FIFO, (MemoryStream) //mX4
22542: 972 uPSI_Int64Em; //Inno
22543: 973 unit uPSI_InstFunc; //Inno
22544: 974 unit uPSI_LibFusion; //Inno
22545: 975 uPSI_SimpleExpression; //Inno
22546: 976 uPSI_unitResourceDetails; //XN
22547: 977 uPSI_unitResFile; //XN
22548: 978 unit uPSI_simpleComport; //mX4
22549: 979 unit uPSI_AfViewershelpers; //Async
22550: 980 unit uPSI_Console; //mX4
22551: 981 unit uPSI_AnalogMeter; //TB
22552: 982 unit uPSI_XPrinter; //TB
22553: 983 unit uPSI_IniFiles; //VCL
22554: 984 unit uPSI_lazIniFiles; //FP
22555: 985 uPSI_testutils; //FP
22556: 986 uPSI_ToolsUnit; (DBTests) //FP
22557: 987 uPSI_fpcunit; //FP
22558: 988 uPSI_testdecorator; //FP
22559: 989 unit uPSI_fpcunittests; //FP
22560: 990 unit uPSI_cTCPBuffer; //Fundamentals 4
22561: 991 unit uPSI_Glut; //Open GL
22562: 992 uPSI_LEDBitmaps; //mX4
22563: 993 uPSI_FileClass; //Inno
22564: 994 uPSI_FileUtilsClass; //mX4
22565: 995 uPSI_ComPortInterface; //Kit //mX4
22566: 996 unit uPSI_SwitchLed; //mX4

```

```

22567: 997 unit uPSI_cyDmmCanvas; //cY
22568: 998 uPSI_uColorFunctions; //DFF
22569: 999 uPSI_uSettings; //DFF
22570: 1000 uPSI_cyDebug.pas //cY
22571: 1001 uPSI_cyColorMatrix; //cY
22572: 1002 unit uPSI_cyCopyFiles; //cY
22573: 1003 unit uPSI_cySearchFiles; //cY
22574: 1004 unit uPSI_cyBaseMeasure; //cY
22575: 1005 unit uPSI_PJStreams; //DD
22576: 1006 unit uPSI_cyRunTimeResize; //cY
22577: 1007 unit uPSI_jcontrolutils; //cY
22578: 1008 unit uPSI_kcMapViewer; (+GEONames) //kc
22579: 1009 unit uPSI_kcMapViewerDESynapse; //kc
22580: 1010 unit uPSI_cparserutils; (+GIS_SysUtils) //kc
22581: 1011 unit uPSI_LedNumber; //TurboPower
22582: 1012 unit uPSI_StStrL; //SysTools
22583: 1013 unit uPSI_IndGnouMeter; //LAZ
22584: 1014 unit uPSI_Sensors; //LAZ
22585: 1015 unit uPSI_pwmain; //cgi of powtils //Pow
22586: 1016 unit uPSI_HTMLUtil; //Pow
22587: 1017 unit uPSI_synwrap1; //httpsend //Pow
22588: 1018 unit StreamWrap1 //Pow
22589: 1019 unit uPSI_pwmain; //Pow
22590: 1020 unit pwtypes //Pow
22591: 1021 uPSI_W32VersionInfo //LAZ
22592: 1022 unit uPSI_IpAnim; //LAZ
22593: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
22594: 1024 unit uPSI_LrtPoTools; //LAZ
22595: 1025 unit uPSI_Laz_DOM; //LAZ
22596: 1026 unit uPSI_hhAvComp; //LAZ
22597: 1027 unit uPSI_GPS2; //LAZ
22598: 1028 unit uPSI_GPS; //LAZ
22599: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo
22600: 1030 unit uPSI_NMEA; // GPS
22601: 1031 unit uPSI_ScreenThreeDLab;
22602: 1032 unit uPSI_Spin; //VCL
22603: 1033 unit uPSI_DynaZip;
22604: 1034 unit uPSI_clockExpert;
22605: 1035 unit debugLn
22606:
22607:
22608:
22609: /////////////////////////////////
22610: //Form Template Library FTL
22611: /////////////////////////////////
22612:
22613: 34 FTL For Form Building out of the Script, eg. 399_form_templates.txt
22614:
22615: 045 unit uPSI_VListView TFormListView;
22616: 263 unit uPSI_JvProfiler32; TProfReport
22617: 270 unit uPSI_ImgList; TCustomImageList
22618: 278 unit uPSI_JvImageWindow; TJvImageWindow
22619: 317 unit uPSI_JvParserForm; TJvHTMLParserForm
22620: 497 unit uPSI_DebugBox; TDebugBox
22621: 513 unit uPSI_ImageWin; TImageForm, TImageForm2
22622: 514 unit uPSI_CustomDrawTreeView; TCustomDrawForm
22623: 515 unit uPSI_GraphWin; TGraphWinForm
22624: 516 unit uPSI_actionMain; TActionForm
22625: 518 unit uPSI_CtlPanel; TAppletApplication
22626: 529 unit uPSI_MDIEdit; TEditForm
22627: 535 unit uPSI_CoolMain; {browser} TWeb MainForm
22628: 538 unit uPSI_frmExportMain; TSynexportForm
22629: 585 unit uPSI_usniffer; {/PortScanForm} TSniffForm
22630: 600 unit uPSI_ThreadForm; TThreadSortForm;
22631: 618 unit uPSI_delphi_arduino_Unit1; TLEDForm
22632: 620 unit uPSI_fplotMain; TfplotForm1
22633: 660 unit uPSI_JvDBQueryParamsForm; TJvQueryParamsDialog
22634: 677 unit uPSI_OptionsFrm; TfrmOptions;
22635: 718 unit uPSI_MonForm; TMonitorForm
22636: 742 unit uPSI_SimplePortMain; TPortForm1
22637: 770 unit uPSI_ovccalc; TOvcCalculator //widget
22638: 810 unit uPSI_DbExcept; TDbEngineErrorDlg
22639: 812 unit uPSI_GL_actorUnit1; TglActorForm1 //OpenGL Robot
22640: 846 unit uPSI_synthttp_daemon; TTCPHttpDaemon, TTCPHttpThrd, TPingThread
22641: 867 unit uPSI_Debug; TfrmDebug
22642: 904 unit uPSI_U_HexView; THexForm2
22643: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain) T0scfrmMain
22644: 959 unit uPSI_AdSelCom; TComSelectForm
22645: 1029 unit uPSI_GPSUDemo; TFDemo
22646: 1031 unit uPSI_ScreenThreeDLab; TFormLab3D
22647:
22648:
22649: ex.:with TEditForm.create(self) do begin
22650:   caption:= 'Template Form Tester';
22651:   FormStyle:= fsStayOnTop;
22652:   with editor do begin
22653:     Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
22654:     SelStart:= 0;
22655:     Modified:= False;

```

```

22656:     end;
22657:   end;
22658:   with TWebMainForm.create(self) do begin
22659:     URLs.Text:= 'http://www.kleiner.ch';
22660:     URLsClick(self); Show;
22661:   end;
22662:   with TSynexportForm.create(self) do begin
22663:     Caption:= 'Synexport HTML RTF tester';
22664:     Show;
22665:   end;
22666:   with TThreadSortForm.create(self) do begin
22667:     ShowModal; free;
22668:   end;
22669:   with TCustomDrawForm.create(self) do begin
22670:     width:=820; height:=820;
22671:     image1.height:= 600; //add properties
22672:     image1.picture.bitmap:= image2.picture.bitmap;
22673:     //SelectionBackground1Click(self) CustomDraw1Click(self);
22674:     Background1.click;
22675:     bitmap1.click; Tile1.click;
22676:     ShowModal;
22677:     Free;
22678:   end;
22679:   with TfplotForm1.Create(self) do begin
22680:     BtnPlotClick(self);
22681:     ShowModal; Free;
22682:   end;
22683:   with TOvcCalculator.create(self) do begin
22684:     parent:= aForm;
22685:     //free;
22686:     setbounds(550,510,200,150);
22687:     displaystr:= 'maXcalc';
22688:   end;
22689:   with THexForm2.Create(self) do begin
22690:     ShowModal;
22691:     Free;
22692:   end;
22693:
22694: function CheckBox: string;
22695: var idHTTP: TIdHTTP;
22696: begin
22697:   result:= 'version not found';
22698:   if IsInternet then begin
22699:     idHTTP:= TIdHTTP.Create(NIL);
22700:     try
22701:       result:= idHTTP.Get(MXVERSIONFILE2);
22702:       result:= result[1]+result[2]+result[3]+result[4]+result[5];
22703:       if result = MBVER2 then begin
22704:         //output.Font.Style:= [fsbold];
22705:         //Speak(' A new Version '+vstr+' of max box is available! ');
22706:         result:= ('!!!! OK. You have latest Version: '+result+ ' available at '+MXSITE);
22707:       end;
22708:       //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
22709:     finally
22710:       idHTTP.Free
22711:     end;
22712:   end;
22713: end;
22714:
22715: function ApWinExecAndWait32(FileName:PChar; CommandLine:PChar; Visibility:Integer):Integer;
22716: function KillTask(ExeFileName: string): Integer;
22717: procedure KillProcess(hWindowHandle: HWND);
22718: function FindWindowByTitle(WindowTitle: string): Hwnd;
22719:
22720:   SendMCICmd('open waveaudio shareable'); //parallels
22721:   SendMCICmd('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\maxbox.wav"');
22722:   SendMCICmd('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\moon.wav"');
22723:   SendMCICmd('close waveaudio');
22724:
22725:
22726:
22727: /////////////////////////////////
22728: All maxbox Tutorials Table of Content 2014
22729: ///////////////////////////////
22730: Tutorial 00 Function-Coding (Blix the Programmer)
22731: Tutorial 01 Procedural-Coding
22732: Tutorial 02 OO-Programming
22733: Tutorial 03 Modular Coding
22734: Tutorial 04 UML Use Case Coding
22735: Tutorial 05 Internet Coding
22736: Tutorial 06 Network Coding
22737: Tutorial 07 Game Graphics Coding
22738: Tutorial 08 Operating System Coding
22739: Tutorial 09 Database Coding
22740: Tutorial 10 Statistic Coding
22741: Tutorial 11 Forms Coding
22742: Tutorial 12 SQL DB Coding
22743: Tutorial 13 Crypto Coding
22744: Tutorial 14 Parallel Coding

```

```

22745: Tutorial 15 Serial RS232 Coding
22746: Tutorial 16 Event Driven Coding
22747: Tutorial 17 Web Server Coding
22748: Tutorial 18 Arduino System Coding
22749: Tutorial 18_3 RGB LED System Coding
22750: Tutorial 19 WinCOM /Arduino Coding
22751: Tutorial 20 Regular Expressions RegEx
22752: Tutorial 21 Android Coding (coming 2013)
22753: Tutorial 22 Services Programming
22754: Tutorial 23 Real Time Systems
22755: Tutorial 24 Clean Code
22756: Tutorial 25 maXbox Configuration I+II
22757: Tutorial 26 Socket Programming with TCP
22758: Tutorial 27 XML & TreeView
22759: Tutorial 28 DLL Coding (available)
22760: Tutorial 29 UML Scripting (2014)
22761: Tutorial 30 Web of Things (2014)
22762: Tutorial 31 Closures (2014)
22763: Tutorial 32 SQL Firebird (coming 2014)
22764: Tutorial 33 Oscilloscope (coming 2015)
22765: Tutorial 34 GPS Navigation (2014)
22766: Tutorial 35 Web Box (available)
22767: Tutorial 36 Unit Testing (coming 2015)
22768: Tutorial 37 API Coding (coming 2015)
22769: Tutorial 38 3D Coding (coming 2015)
22770: Tutorial 39 GEO Map Coding (available)
22771: Tutorial 39_1 GEO Map Layers Coding (available)
22772: Tutorial 40 REST Coding (coming 2015)
22773:
22774:
22775: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
22776: using Docu for this file is maxbox_functions_all.pdf
22777: PEP - Pascal Education Program Low Lib Lab ShellHell
22778:
22779: http://stackoverflow.com/tags/pascalscript/hot
22780: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
22781: http://sourceforge.net/projects/maxbox #locs:51620
22782: http://sourceforge.net/apps/mediawiki/maxbox
22783: http://www.blaisepascal.eu/
22784: https://github.com/maxkleiner/maxbox3.git
22785: http://www.heise.de/download/maxbox-1176464.html
22786: http://www.softpedia.com/get/Programming/Other-Programming-Files/maxbox.shtml
22787: https://www.facebook.com/pages/Programming-maxBox/166844836691703
22788: http://www.softwareschule.ch/arduino_training.pdf
22789: http://www.delphairea.com
22790: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
22791: http://entwickler-konferenz.de/2014/speakers/max-kleiner
22792:
22793:
22794: All maXbox Examples List
22795: https://github.com/maxkleiner/maxBox3/releases
22796: ****
22797: 000_pas_baseconvert.txt 282_fadengraphik.txt
22798: 000_pas_baseconvert.txt_encrypt 283_SQL_API_messagetimeout.txt
22799: 000_pas_baseconvert.txt_decrypt 284_SysTools4.txt
22800: 001_1_pas_functest - Kopie.txt 285_MineForm_GR32.TXT
22801: 001_1_pas_functest.txt 285_MineForm_GR32main.TXT
22802: 001_1_pas_functest2.txt 285_MineForm_GR32 mainsolution.TXT
22803: 001_1_pas_functest_clx2.txt 285_MineForm_propas.TXT
22804: 001_1_pas_functest_clx2_2.txt 285_MineForm_propas2.TXT
22805: 001_1_pas_functest_openarray.txt 285_minesweeper2.TXT
22806: 001_pas_lottogen.txt 285_Patterns_process.txt
22807: 001_pas_lottogen_template.txt 286_colormixer_jpeg_charcounter.txt
22808: 001_pas_lottogen.txtcopy 286_colormixer_jpeg_charcounter2.txt
22809: 002_pas_russianroulette.txt 287_eventhandling.txt
22810: 002_pas_russianroulette.txtcopy 287_eventhandling2.txt
22811: 002_pas_russianroulette.txtcopy_decrypt 287_eventhandling2_negpower.txt
22812: 002_pas_russianroulette.txtcopy_encrypt 288_bitblt.txt
22813: 003_pas_motion.txt 288_bitblt_resize.txt
22814: 003_pas_motion.txtcopy 289_regression.txt
22815: 004_pas_search.txt 289_regression2.txt
22816: 004_pas_search_replace.txt 290_bestofbox.txt
22817: 004_search_replace_allfunctionlist.txt 290_bestofbox2.txt
22818: 005_pas_oodesign.txt 290_bestofbox3.txt
22819: 005_pas_shelllink.txt 291_3sort_visual_thread.txt
22820: 006_pas_oobatch.txt 292_refactoring2.txt
22821: 007_pas_streamcopy.txt 293_bold_utils.txt
22822: 008_EINMALEINS_FUNC.TXT 293_ib_utils.txt
22823: 008_explanation.txt 293_ib_utils_timetest.txt
22824: 008_pas_verwechselt.txt 294_maxcalc_demo.txt
22825: 008_pas_verwechselt_ibz_bern_func.txt 294_maxcalc_demo2.txt
22826: 008_stack_ibz.TXT 295_easter_calendar.txt
22827: 009_pas_umrunner.txt 295_easter_calendar2.txt
22828: 009_pas_umrunner_all.txt 295_easter_combobox.txt
22829: 009_pas_umrunner_componenttest.txt 297_atomimage.txt
22830: 009_pas_umrunner_solution.txt 297_atomimage2.txt
22831: 009_pas_umrunner_solution_2step.txt 297_atomimage3.txt
22832: 010_pas_oodesign_solution.txt 297_atomimage4.txt
22833: 011_pas_puzzlepas_defect.txt 297_maxonmotor.TXT

```

```

22834: 012_pas_umrunner_solution.txt          297_maxon_atomimage9.txt
22835: 012_pas_umrunner_solution2.txt        298_bitblt_animation.txt
22836: 013_pas_linenumber.txt                298_bitblt_animation2.txt
22837: 014_pas_primetest.txt                 298_bitblt_animation3.txt
22838: 014_pas_primetest_first.txt           298_bitblt_animation4.txt
22839: 014_pas_primetest_sync.txt            298_bitblt_animation4_screensaver.txt
22840: 015_pas_designbycontract.txt          298_bitblt_animation5_screensaver.txt
22841: 015_pas_designbycontract_solution.txt 299_animation.txt
22842: 016_pas_searchrec.txt                 299_animationmotor_arduino.txt
22843: 017_chartgen.txt                     299_animation_formprototype.txt
22844: 018_data_simulator.txt               299_realtimeclock_arduino.txt
22845: 019_dez_to_bin.txt                   299_realtimeclock_arduino2.txt
22846: 019_dez_to_bin_grenzwert_ibz.txt     300_treeview.txt
22847: 020_proc_feedback.txt                300_treeview_test.txt
22848: 021_pas_symkey.txt                  300_treeview_test2.txt
22849: 021_pas_symkey_solution.txt          300_treeview_test3.txt
22850: 022_pas_filestreams.txt              301_LED_Arduino3.txt
22851: 023_pas_find_searchrec.txt            301_led_arduino3_simple.txt
22852: 023_pas_pathfind.txt                301_led_arduino3_simplecode.txt
22853: 024_pas_TFileStream_records.txt      301_log_arduino.txt
22854: 025_prime_direct.txt                301_log_arduino2.txt
22855: 026_pas_memorystream.txt             301_SQL_DBfirebird3.txt
22856: 027_pas_shellexecute_beta.txt       301_SQL_DBfirebird4.txt
22857: 027_pas_shellexecute_solution.txt   302_LCLActivity_java.txt
22858: 028_pas_dataset.txt                 302_LED_DataLogger.txt
22859: 029_pas_assignfile.txt              303_Android_LCLActivity_java.txt
22860: 029_pas_assignfile_dragndropexe.txt 303_webserver.txt
22861: 030_palindrome_2.txt                303_webserver2.txt
22862: 030_palindrome_tester.txt            303_webserver_alldocs2.txt
22863: 030_pas_recursion.txt               303_webserver_alldocs2_tester.txt
22864: 030_pas_recursion2.txt              303_webserver_minimal.txt
22865: 031_pas_hashcode.txt                303_webserver_simple.txt
22866: 032_pas_crc_const.txt              304_st_system.txt
22867: 033_pas_cipher.txt                 305_indy_elizahttpserver.TXT
22868: 033_pas_cipher_def.txt             305_indy_elizahttpserver2.TXT
22869: 033_pas_cipher_file_2_solution.txt 305_indy_elizahttpserver3.TXT
22870: 034_pas_soundbox.txt                305_indy_elizahttpserver4file.TXT
22871: 035_pas_crcscript.txt              305_webserver_arduino.txt
22872: 035_pas_CRCscript_modbus.txt        305_webserver_arduino2.txt
22873: 036_pas_includetest.txt            305_webserver_arduino3.txt
22874: 036_pas_includetest_basta.txt       305_webserver_arduino3ibz.txt
22875: 037_pas_define_demo32.txt          305_webserver_arduino3ibz_rgb_led.txt
22876: 038_pas_box_demonstrator.txt       305_webserver_arduino3test.txt
22877: 039_pas_dllcall.txt                306_SPS_http_command.txt
22878: 040_paspointer.txt                 307_all_booleanlogic.txt
22879: 040_paspointer_old.txt             308_bitbox3.txt
22880: 041_pasplotter.txt                 308_bitbox3_exec.txt
22881: 041_pasplotter_plus.txt            308_boolean_animation.txt
22882: 042_pas_kgv_ggt.txt                308_boolean_animation2.txt
22883: 043_pas_proceduretype.txt          309_regex_power.txt
22884: 044_pas_l4queens_solwith14.txt     309_regex_powertester2.txt
22885: 044_pas_8queens.txt                309_regex_powertester3.txt
22886: 044_pas_8queens_sol2.txt            310_regex_decorator.TXT
22887: 044_pas_8queens_solutions.txt      312_ListView.txt
22888: 044_queens_performer.txt           313_dmath_dll.txt
22889: 044_queens_performer2.txt           314_fundamentals4_tester.TXT
22890: 044_queens_performer2tester.txt     315_funcplot_dmath.TXT
22891: 045_pas_listhandling.txt            316_fileutils_cdatetime_tester.TXT
22892: 046_pas_records.txt                 317_excel_export_tester.TXT
22893: 047_pas_modulal0.txt                318_excel_export.TXT
22894: 048_pas_romans.txt                 318_excel_export2.TXT
22895: 049_pas_ifdemo.txt                 318_excel_export3.TXT
22896: 049_pas_ifdemo_BROKER.txt          318_excel_export3_tester.TXT
22897: 050_pas_primetest2.txt              319_superfunctions_math.TXT
22898: 050_pas_primetester_thieves.txt    319_superfunctions_mathdefect.TXT
22899: 050_program_starter.txt             320_superfunctions.TXT
22900: 050_program_starter_performance.txt 320_superfunctions2.TXT
22901: 051_pas_findtext_solution.txt       321_SQL_Excel.txt
22902: 052_pas_text_as_stream.txt          321_SQL_Excel2.txt
22903: 052_pas_text_as_stream_include.txt 321_SQL_Excel_Export.txt
22904: 053_pas_singleton.txt              321_SQL_ExportExec.txt
22905: 054_pas_speakpassword.txt           321_SQL_ExportTest.txt
22906: 054_pas_speakpassword2.txt           321_SQL_SAS_tester3.txt
22907: 054_pas_speakpassword_searchtest.txt 321_SQL_SAS_tester3_selfcompile.txt
22908: 055_pas_factorylist.txt             321_SQL_SAS_tester3_selfcompile2.txt
22909: 056_pas_demeter.txt                 321_SQL_SAS_tester4.txt
22910: 057_pas_dirfinder.txt                321_SQL_SAS_updater.txt
22911: 058_pas_filefinder.txt              322_timezones.TXT
22912: 058_pas_filefinder_pdf.txt          323_datefind_fulltext_search.txt
22913: 058_pas_filefinder_screview.txt     323_datefind_fulltext_searchtester.txt
22914: 058_pas_filefinder_screview2.txt    324_interfacenavi.TXT
22915: 058_pas_filefinder_screview3.txt    325_ampelsteuerung.txt
22916: 059_pas_timertest.txt                325_analogclock.txt
22917: 059_pas_timertest_2.txt              326_world_analogclock.txt
22918: 059_pas_timertest_time_solution.txt 326_world_analogclock2.txt
22919: 059_timerobject_starter2.txt         327_atomimage_clock.txt
22920: 059_timerobject_starter2_ibz2_async.txt 328_starfield.txt
22921: 059_timerobject_starter2_uml.txt     329_starfield2.txt
22922: 059_timerobject_starter2_uml_main.txt 330_myclock.txt

```

```

22923: 059_timerobject_starter4_ibz.txt          330_myclock2.txt
22924: 060_pas_datefind.txt                     331_SQL_DBfirebird4.txt
22925: 060_pas_datefind_exceptions2.txt        332_jprofiler.txt
22926: 060_pas_datefind_exceptions_CHECKTEST.txt 332_jprofiler_form.txt
22927: 060_pas_datefind_fulltext.txt           332_jprofiler_form2.txt
22928: 060_pas_datefind_plus.txt                333_querybyexample.txt
22929: 060_pas_datefind_plus_mydate.txt         333_querybyexample2.txt
22930: 061_pas_randomwalk.txt                  334_jutils_u.txt
22931: 061_pas_randomwalk_plus.txt             335_atomimage5.txt
22932: 062_paskorrelation.txt                 335_atomimage6.txt
22933: 063_pas_calculateform.txt              335_atomimage7.txt
22934: 063_pas_calculateform_2list.txt        336_digiclock.txt
22935: 064_pas_timetest.txt                   336_digiclock2.txt
22936: 065_pas_bitcounter.txt                 336_digiclock2test.txt
22937: 066_pas_eliza.txt                      336_digiclock3.txt
22938: 066_pas_eliza_include_sol.txt          337_4games.txt
22939: 067_pas_morse.txt                     337_4games_inone.txt
22940: 068_pas_piezo_sound.txt                338_compress.txt
22941: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT   338_compress2.txt
22942: 069_my_LEDBOX.TXT                     339_ntfs.txt
22943: 069_pas_ledmatrix.txt                 340_docudate.txt
22944: 069_pas_LEDMATRIX_Alphabet.txt        340_logsimulation.txt
22945: 069_pas_LEDMATRIX_Alphabet_run.txt    340_logsimulation2.txt
22946: 069_pas_LEDMATRIX_Alphabet_tester.txt 340_soundControltype.txt
22947: 069_PAS_LEDMATRIX_COLOR.TXT            341_blix_clock.txt
22948: 069_pas_ledmatrix_fixedit.txt          341_blix_clock2.txt
22949: 069_pas_LEDMATRIX_soundbox.txt         341_blix_clock_tester.txt
22950: 069_pas_LEDMATRIX_soundbox2.txt        342_set Enumerator.txt
22951: 069_Richter_MATRIX.TXT                343_dice2.txt
22952: 070_pas_functionplot.txt               344_pe_header.txt
22953: 070_pas_functionplotter2.txt           344_pe_header2.txt
22954: 070_pas_functionplotter2_mx4.txt       345_velocity.txt
22955: 070_pas_functionplotter2_tester.txt    346_conversions.txt
22956: 070_pas_functionplotter3.txt           347_picturereview.txt
22957: 070_pas_functionplotter4.txt           348_duallistview.txt
22958: 070_pas_functionplotter_digital.txt    349_biginteger.txt
22959: 070_pas_functionplotter_elliptic.txt  350_parserform.txt
22960: 070_pas_function_helmholtz.txt        351_chartform.txt
22961: 070_pas_textcheck_experimental.txt     351_chartform2.txt
22962: 071_pas_graphics.txt                  351_chartform3.txt
22963: 071_pas_graphics_drawsym.txt          352_array_unittest.txt
22964: 071_pas_graphics_drawsym_save.txt     353_smtp_email.txt
22965: 071_pas_graphics_random.txt           353_smtp_email2.txt
22966: 072_pas_fractals.txt                  354_josephus.txt
22967: 072_pas_fractals_2.txt                355_life_of_PI.txt
22968: 072_pas_fractals_blackhole.txt         356_3D_printer.txt
22969: 072_pas_fractals_performace.txt        357_fplot.TXT
22970: 072_pas_fractals_perfomance_new.txt   358_makesound.txt
22971: 072_pas_fractals_perfomance_sharp.txt 359_charsetrules.TXT
22972: 072_pas_fractals_performance.txt       360_allobjects.TXT
22973: 072_pas_fractals_performance_mx4.txt   360_JvPaintFX.TXT
22974: 073_pas_forms.txt                     361_heartbeat_wave.TXT
22975: 074_pas_chartgenerator.txt             362_maxonmotor2.TXT
22976: 074_pas_chartgenerator_solution.txt   363_compress_services.txt
22977: 074_pas_chartgenerator_solution_back.txt 363_compress_services2.txt
22978: 074_pas_charts.txt                    364_pdf_services.txt
22979: 075_bitmap_Artwork2.txt               365_memorystream.txt
22980: 075_pas_bitmappuzzle.txt              365_memorystream2.txt
22981: 075_pas_bitmappuzzle24.prod.txt       365_memorystream_test.txt
22982: 075_pas_bitmappuzzle2_prod.txt         365_U_HexView.txt
22983: 075_pas_bitmappuzzle3.txt              366_mp3player.txt
22984: 075_pas_bitmapsolve.txt               366_mp3player2.txt
22985: 075_pas_bitmap_Artwork.txt             366_mp3player2_themestest.txt
22986: 075_pas_puzzlespas_solution.txt       367_silvi_player_widgets.txt
22987: 076_pas_3dcube.txt                    367_silvi_player_widgets2.txt
22988: 076_pas_circle.txt                   367_widgets.txt
22989: 077_pas_mmshow.txt                   368_configuration_demo.txt
22990: 078_pas_pi.txt                       369_macro_demo.txt
22991: 079_pas_3dcube_animation.txt          370_callback2grid.TXT
22992: 079_pas_3dcube_animation4.txt         370_richedit.txt
22993: 079_pas_3dcube_plus.txt              370_richedit_highlight.txt
22994: 080_pas_hanoi.txt                   370_syndit.txt
22995: 080_pas_hanoi2.txt                  370_syndit2.txt
22996: 080_pas_hanoi2_file.txt              370_syndit2_mxtester.txt
22997: 080_pas_hanoi2_sol.txt              370_syndit2_mxtester2.txt
22998: 080_pas_hanoi2_tester.txt            371_maxbook_v4tester.txt
22999: 080_pas_hanoi2_tester_fast.txt       372_stackibz2_memoryalloc.TXT
23000: 080_pas_hanoi3.txt                  372_syndit_export.txt
23001: 081_pas_chartist2.txt               373_batman.txt
23002: 082_pas_biorythmus.txt              373_fractals_tvout.txt
23003: 082_pas_biorythmus_solution.txt     374_realtime_random.txt
23004: 082_pas_biorythmus_solution_3.txt   374_realtime_random2.txt
23005: 082_pas_biorythmus_test.txt          374_realtime_randomtest.txt
23006: 083_pas_GITARRE.txt                374_realtime_randomtest2.txt
23007: 083_pas_soundbox_tones.txt           375_G9_musicbox.txt
23008: 084_pas_waves.txt                  376_collections_list.txt
23009: 085_mxsinus_logo.txt                377_simpleXML.txt
23010: 085_sinus_plot_waves.txt            377_smartXML.txt
23011: 086_pas_graph_arrow_heart.txt       377_smartXMLWorkshop.txt

```

23012: 087_bitmap_loader.txt	377_smartXMLWorkshop2.txt
23013: 087_pas_bitmap_solution.txt	378_queryperformance3.txt
23014: 087_pas_bitmap_solution2.txt	378_REST1.txt
23015: 087_pas_bitmap_subimage.txt	378_REST2.txt
23016: 087_pas_bitmap_test.txt	379_timefunc.txt
23017: 088_pas_soundbox2_mp3.txt	379_timefuncTesterfilemon.txt
23018: 088_pas_soundbox_mp3.txt	380_coolfunc.txt
23019: 088_pas_sphere_2.txt	380_coolfunc2.txt
23020: 089_pas_gradient.txt	380_coolfunc_tester.txt
23021: 089_pas_maxland2.txt	381_bitcoin_simulation.txt
23022: 090_pas_sudoku4.txt	382_GRMath.TXT
23023: 090_pas_sudoku4_2.txt	382_GRMath_PI_Proof.TXT
23024: 091_pas_cube4.txt	382_GRMath_Riemann.TXT
23025: 092_pas_statistics4.txt	383_MDAC_DCOM.txt
23026: 093_variance.txt	384_TeamViewerID.TXT
23027: 093_variance_debug.txt	386_InternetRadio.TXT
23028: 094_pas_daysold.txt	387_fulltextfinder.txt
23029: 094_pas_stat_date.txt	387_fulltextfinder_cleancode.txt
23030: 095_pas_ki_simulation.txt	387_fulltextfinder_fast.txt
23031: 096_pas_geisen_problem.txt	387_fulltext_getscripttest.txt
23032: 096_pas_montyhall_problem.txt	388_TCPServerSock.TXT
23033: 097_lotto_proofofconcept.txt	388_TCPServerSock2.TXT
23034: 097_pas_lottocombinations_beat_plus.txt	388_TCPServerSockClient.TXT
23035: 097_pas_lottocombinations_beat_plus2.txt	389_TAR_Archive.TXT
23036: 097_pas_lottocombinations_universal.txt	389_TAR_Archive_test.TXT
23037: 097_pas_lottosimulation.txt	389_TAR_Archive_test2.TXT
23038: 098_pas_chartgenerator_plus.txt	390_Callback3.TXT
23039: 099_pas_3D_show.txt	390_Callback3Rec.TXT
23040: 200_big_numbers.txt	390_CallbackClean.TXT
23041: 200_big_numbers2.txt	390_StringlistHTML.TXT
23042: 201_streamload_xml.txt	391_ToDo_List.TXT
23043: 202_systemcheck.txt	392_Barcodex.TXT
23044: 203_webservice_simple_intftester.txt	392_Barcodex2.TXT
23045: 204_webservice_simple.txt	392_Barcodex23.TXT
23046: 205_future_value_service.txt	392_Barcodex2scholz.TXT
23047: 206_DTD_string_functions.txt	392_Barcodex3scholz.TXT
23048: 207_ibz2_async_process.txt	393_QRCode.TXT
23049: 208_crc32_hash.txt	393_QRCode2.TXT
23050: 209_cryptohash.txt	393_QRCode2Direct.TXT
23051: 210_public_private.txt	393_QRCode2DirectIndy.TXT
23052: 210_public_private_cryptosystem.txt	393_QRCode2Direct_detlef.TXT
23053: 211_wipe_pattern.txt	393_QRCode3.TXT
23054: 211_wipe_pattern2.txt	394_networkgraph.TXT
23055: 211_wipe_pattern_solution.txt	394_networkgraph_depwalkutilstest.TXT
23056: 212_pas_statisticmodule4.TXT	394_networkgraph_depwalkutilstest2.TXT
23057: 212_pas_statisticmoduled.txt	395_USBController.TXT
23058: 212_statisticmodule4.txt	396_Sort.TXT
23059: 213_pas_BBP_Algo.txt	397_Hotlog.TXT
23060: 214_mxdocudemo.txt	397_Hotlog2.TXT
23061: 214_mxdocudemo2.txt	398_ustrings.txt
23062: 214_mxdocudemo3.txt	399_form_templates.txt
23063: 215_hints_test.TXT	400_fploottchart.TXT
23064: 216_warnings_test.TXT	400_fploottchart2.TXT
23065: 217_pas_heartbeat.txt	400_fploottchart2teetest.TXT
23066: 218_biorhythmus_studio.txt	400_QRCodeMarket.TXT
23067: 219_cipherbox.txt	401_tfilerun.txt
23068: 219_crypt_source_comtest_solution.TXT	402_richedit2.txt
23069: 220_cipherbox_form.txt	403_outlookspy.txt
23070: 220_cipherbox_form2.txt	404_simplebrowser.txt
23071: 221_bcd_explain.txt	405_datefinder_today.txt
23072: 222_memoform.txt	406_portscan.txt
23073: 223_directorybox.txt	407_indydemo.txt
23074: 224_dialogs.txt	408_testroboter.txt
23075: 225_sprite_animation.txt	409_excel_control.txt
23076: 226_ASCII_Grid2.TXT	410_keyboardevent.txt
23077: 227_animation.txt	411_json_test.txt
23078: 227_animation2.txt	412_Zeosutils.txt
23079: 228_android_calendar.txt	413_listview2.txt
23080: 229_android_game.txt	414_avrdude_flash.txt
23081: 229_android_game_tester.txt	415_avrdude_writehex.txt
23082: 230_DataProvider.txt	416_sonar_startscriptEKON.TXT
23083: 230_DataSetProvider.txt	416_sonar_startscriptEKON_reporting.TXT
23084: 230_DataSetXMLBackupScholz.txt	417_GRMath_PI_Proof2.TXT
23085: 231_DBGrid_access.txt	418_functional_paradigm.txt
23086: 231_DBGrid_XMLaccess.txt	419_archimedes_spiral.txt
23087: 231_DBGrid_XMLaccess2.txt	419_archimedes_spiral2.txt
23088: 231_DBGrid_XMLaccess_locatetester.txt	420_archimedes_arduino.txt
23089: 231_DBGrid_XML_CDS_local.txt	420_Lissajous.txt
23090: 232_outline.txt	421_PI_Power.TXT
23091: 232_outline_2.txt	421_PI_Power2.TXT
23092: 233_modular_form.txt	422_world_bitboxx.txt
23093: 234_debugoutform.txt	423_game_of_life.txt
23094: 235_fastform.TXT	423_game_of_life2.txt
23095: 236_componentpower.txt	423_game_of_life3.txt
23096: 236_componentpower_back.txt	423_game_of_life3_test.TXT
23097: 237_pas_4forms.txt	423_game_of_life4.txt
23098: 238_lottogen_form.txt	423_game_of_life4_kryptum.TXT
23099: 239_pas_sierpinski.txt	424_opengl_tester.txt
23100: 239_pas_sierpinski2.txt	425_reversi_game.txt

```

23101: 240_unitGlobal_tester.txt          426_IBUtils.TXT
23102: 241_db3_sql_tutorial.txt          427_IBDatabase.TXT
23103: 241_db3_sql_tutorial2.txt         428_SortGrid.TXT
23104: 241_db3_sql_tutorial2fix.txt      429_fileclass.txt
23105: 241_db3_sql_tutorial3.txt         430_fileoperation.txt
23106: 241_db3_sql_tutorial3connect.txt  430_fileoperation_tester.txt
23107: 241_db3_sql_tutorial3_ftest.txt   431_performance_index.txt
23108: 241_RTL_SET2.txt                 432_shortstring_routines.txt
23109: 241_RTL_SET2_tester.txt          433_video_avicap.txt
23110: 242_Component_Control.txt        433_video_avicap2.txt
23111: 243_tutorial_loader.txt          434_GSM_module.TXT
23112: 244_script_loader_loop.txt       435_httpcommon.txt
23113: 245_formapp2.txt                436_GraphicSplitter.txt
23114: 245_formapp2_tester.txt          436_GraphicSplitter_form.txt
23115: 245_formapp2_testerX.txt         436_GraphicSplitter_form2.txt
23116: 246_httpapp.txt                 436_teetest_screen.TXT
23117: 247_datecalendar.txt            436_teetest_screen2.TXT
23118: 248_ASCII_Grid2_sorted.TXT       437_WinAPItop.txt
23119: 249_picture_grid.TXT            437_WinAPItop_Firebirdtester.txt
23120: 250_tipsandtricks2.txt          438_OvcInternational.txt
23121: 250_tipsandtricks3.txt          439_AsyncFreeDemo.txt
23122: 250_tipsandtricks3api.txt       439_AsyncFreeDemoForm.txt
23123: 250_tipsandtricks3_admin_elevation.txt 440_DLL_Tutor.txt
23124: 250_tipsandtricks3_tester.txt    440_DLL_Tutor2.txt
23125: 250_tipsandtricks4_tester.txt    440_XML_Tutor.txt
23126: 250_tipsandtricks4_tester2.txt   440_XML_Tutor2.txt
23127: 251_compare_noise_gauss.txt     441_make_app.txt
23128: 251_whitenoise.txt              442_arduino_rgb_led.txt
23129: 251_whitenoise2.txt             443_webserver_arduino_rgb_light.txt
23130: 252_hilbert_turtle.txt           443_webserver_arduino_rgb_light4.txt
23131: 252_pas_hilbert.txt             444_webserver_arduino3ibz_rgb_led_basta.txt
23132: 253_opearatingsystem3.txt        445_datagrid.txt
23133: 254_dynarrays.txt               445_datagrid2.txt
23134: 255_einstein.txt                445_datagrid_android_arduino.txt
23135: 256_findconsts_of_EXE.txt       446_arduino_timer.txt
23136: 256_findfunctions2_of_EXE.txt   447_patternFrm_mx3.txt
23137: 256_findfunctions2_of_EXBawerp.txt 448_Synapse.txt
23138: 256_findfunctions2_of_EXEspec.txt 448_Synapse2.txt
23139: 256_findfunctions3.txt          449_dweb_start_tester.txt
23140: 256_findfunctions_of_EXE.txt    450_Synapse_HTTPS.txt
23141: 257_AES_Cipher.txt              450_Synapse_Mime.txt
23142: 258_AES_cryptobox.txt           450_Synapse_ScanPing.txt
23143: 258_AES_cryptobox2.txt          451_ocx_player.txt
23144: 258_AES_cryptobox2_passdlg.txt 451_OCX_WinPlayer2.txt
23145: 259_AES_crypt_directory.txt     452_dbtreeview.txt
23146: 260_sendmessage_2.TXT          452_dbtreeview2.txt
23147: 260_sendmessage_beta.TXT       453_stdfuncs.txt
23148: 261_probability.txt            454_fileStream.txt
23149: 262_mxoutputdemo4.txt          455_functionfun.txt
23150: 263_async_sound.txt            455_functionfun2.txt
23151: 264_vclutils.txt               455_functionfun2_test.txt
23152: 264_VCL_utils2.txt             457_ressource_grid.txt
23153: 265_timer_API.txt              458_atomimageX.txt
23154: 266_serial_interface.txt       459_cindyfunc.txt
23155: 266_serial_interface2.txt      459_cindyfunc2.txt
23156: 266_serial_interface3.txt      460_TopTenFunctions.txt
23157: 267_ackermann_rec.txt          461_sqlform_calwin.txt
23158: 267_ackermann_variants.txt     462_caesarcipher.txt
23159: 268_DBGrid_tree.txt            463_global_exception.txt
23160: 269_record_grid.TXT            464_function_procedure.txt
23161: 270_Jedi_FunctionPower.txt     464_function_procedure2.txt
23162: 270_Jedi_FunctionPowerTester.txt 464_function_procedure3.txt
23163: 271_closures_study.txt         465_U_HexView.txt
23164: 271_closures_study_workingset2.txt 466_moon.txt
23165: 272_pas_function_show.txt      466_moon_inputquery.txt
23166: 273_pas_function_show2.txt     467_cardmagic.txt
23167: 274_library_functions.txt      467_helmholtz_graphic.txt
23168: 275_turtle_language.txt         468_URLMon.txt
23169: 275_turtle_language_save.txt   468_URLMon2.txt
23170: 276_save_algo.txt              469_formarrow.txt
23171: 276_save_algo2.txt             469_formarrow_datepicker.txt
23172: 277_functionsfor39.txt         469_formarrow_datepicker_ibz_result.txt
23173: 278_DB_Dialogs.TXT            469_ibzresult.txt
23174: 279_hexer2.TXT                470_DFFUtils_compiled.txt
23175: 279_hexer2macro.TXT            470_DFFUtils_ScrollingLED.txt
23176: 279_hexer2macroback.TXT        470_Oscilloscope.txt
23177: 280_UML_process.txt            470_Oscilloscope_code.txt
23178: 280_UML_process_knabe2.txt     471_cardmagic.txt
23179: 280_UML_process_knabe3.txt     471_cardmagic2.txt
23180: 280_UML_process_TIM_Botzenhardt.txt 472_allcards.txt
23181: 280_UML_TIM_Seitz.txt          473_comboiset.txt
23182: 281_picturepuzzle.txt          474_wakeonlan.txt
23183: 281_picturepuzzle2.txt         474_wakeonlan2.txt
23184: 281_picturepuzzle3.txt          476_getscripttest.txt
23185: 281_picturepuzzle4.txt          477_filenameonly.txt
23186: 479_inputquery.txt              480_regex_pathfinder.txt
23187: 480_regex_pathfinder2.txt       481_processList.txt
23188: 482_processPipe.txt             482_processPipeGCC.txt
23189: 483_PathFuncTest_mx.txt         484_filefinder3.txt

```

```

23190: 485_InnoFunc.txt
23191: 487_asyncKeyState.txt
23192: 489_simpleComport.txt
23193: 491_analogmeter.txt
23194: 493_gadgets.txt
23195: 496_InstallX.txt
23196: 498_UnitTesting.txt
23197: 500_diceoflives.txt
23198: 502_findalldocs.txt
23199: 504_fileclass.txt
23200: 506_colormatrix.txt
23201: 508_simplecomportmorse.txt
23202: 509_509_GEOMap2_SReverse.TXT
23203: 511_LEDLabel.txt
23204: 513_StreamIntegration.txt
23205: 515_ledclock3.txt
23206: 517_animation7.txt
23207: 519_powtils.txt
23208: 521_iputils2.txt
23209: 523_NMEA.txt
23210: 525_GEO84s.txt
23211: 527_GPSDemo.txt
23212: 529_profilertest.txt
23213: 531_profilertest.txt
23214:
23215:
23216: Web Script Examples:
23217:
23218: http://www.softwareschule.ch/examples/performer.txt';
23219: http://www.softwareschule.ch/examples/turtle.txt';
23220: http://www.softwareschule.ch/examples/SQLExport.txt';
23221: http://www.softwareschule.ch/examples/Richter.txt';
23222: http://www.softwareschule.ch/examples/checker.txt';
23223: http://www.softwareschule.ch/examples/demoscript.txt';
23224: http://www.softwareschule.ch/examples/ibzresult.txt';
23225: http://www.softwareschule.ch/examples/performindex.txt
23226: http://www.softwareschule.ch/examples/processlist.txt
23227: http://www.softwareschule.ch/examples/game.txt
23228: http://www.softwareschule.ch/examples/GEOGPS.txt
23229: http://www.softwareschule.ch/examples/turtle2.txt
23230:
23231:
23232:
23233: Delphi Basics Run Time Library listing
23234: ****
23235: A
23236: Compiler Directive $A Determines whether data is aligned or packed
23237: Compiler Directive $Align Determines whether data is aligned or packed
23238: Compiler Directive $AppType Determines the application type : GUI or Console
23239: Procedure SysUtils Abort Aborts the current processing with a silent exception
23240: Function System Abs Gives the absolute value of a number (-ve sign is removed)
23241: Directive Abstract Defines a class method only implemented in subclasses
23242: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
23243: Function System Addr Gives the address of a variable, function or procedure
23244: Keyword And Boolean and or bitwise and of two arguments
23245: Type System AnsiChar A character type guaranteed to be 8 bits in size
23246: Function SysUtils AnsiCompareStr Compare two strings for equality
23247: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
23248: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
23249: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
23250: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
23251: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
23252: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
23253: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
23254: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string
23255: Function StrUtils AnsiPos Find the position of one string in another
23256: Function StrUtils AnsiReplaceStr Replaces a part of one string with another
23257: Function StrUtils AnsiReverseString Reverses the sequence of letters in a string
23258: Function StrUtils AnsiRightStr Extracts characters from the right of a string
23259: Function StrUtils AnsiStartsStr Returns true if a string starts with a substring
23260: Type System AnsiString A data type that holds a string of AnsiChars
23261: Function SysUtils AnsiUpperCase Change lower case characters in a string to upper case
23262: Procedure System Append Open a text file to allow appending of text to the end
23263: Procedure SysUtils AppendStr Concatenate one string onto the end of another
23264: Function Math ArcCos The Arc Cosine of a number, returned in radians
23265: Function Math ArcSin The Arc Sine of a number, returned in radians
23266: Function System ArcTan The Arc Tangent of a number, returned in radians
23267: Keyword Array A data type holding indexable collections of data
23268: Keyword As Used for casting object references
23269: Procedure System Assign Assigns a file handle to a binary or text file
23270: Function System Assigned Returns true if a reference is not nil
23271: Procedure System AssignFile Assigns a file handle to a binary or text file
23272: Procedure Printers AssignPrn Treats the printer as a text file - an easy way of printing text
23273:
23274: B
23275: Compiler Directive $B Whether to short cut and and or operations
23276: Compiler Directive $BoolEval Whether to short cut and and or operations
23277: Procedure SysUtils Beep Make a beep sound
23278: Keyword Begin Keyword that starts a statement block

```

```

23279: Function System BeginThread Begins a separate thread of code execution
23280: Procedure System BlockRead Reads a block of data records from an untyped binary file
23281: Procedure System BlockWrite Writes a block of data records to an untyped binary file
23282: Type System Boolean Allows just True and False values
23283: Function Classes Bounds Create a TRect value from top left and size values
23284: Procedure System Break Forces a jump out of a single loop
23285: Type System Byte An integer type supporting values 0 to 255
23286:
23287: C
23288: Type System Cardinal The basic unsigned integer type
23289: Keyword Case A mechanism for acting upon different values of an Ordinal
23290: Function StdConvns CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
23291: Function SysUtils ChangeFileExt Change the extension part of a file name
23292: Type System Char Variable type holding a single character
23293: Procedure System ChDir Change the working drive plus path for a specified drive
23294: Function System Chr Convert an integer into a character
23295: Keyword Class Starts the declaration of a type of object class
23296: Procedure System Close Closes an open file
23297: Procedure System CloseFile Closes an open file
23298: Variable System CmdLine Holds the execution text used to start the current program
23299: Type System Comp A 64 bit signed integer
23300: Function SysUtils CompareStr Compare two strings to see which is greater than the other
23301: Function SysUtils CompareText Compare two strings for equality, ignoring case
23302: Function Math CompareValue Compare numeric values with a tolerance
23303: Function System Concat Concatenates one or more strings into one string
23304: Keyword Const Starts the definition of fixed data values
23305: Keyword Constructor Defines the method used to create an object from a class
23306: Procedure System Continue Forces a jump to the next iteration of a loop
23307: Function ConvUtils Convert Convert one measurement value to another
23308: Function System Copy Create a copy of part of a string or an array
23309: Function System Cos The Cosine of a number
23310: Function SysUtils CreateDir Create a directory
23311: Type System Currency A floating point type with 4 decimals used for financial values
23312: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
23313: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
23314: Variable SysUtils CurrencyString The currency string used in currency display functions
23315: Function SysUtils CurrToStr Convert a currency value to a string
23316: Function SysUtils CurrToStrF Convert a currency value to a string with formatting
23317:
23318: D
23319: Compiler Directive $D Determines whether application debug information is built
23320: Compiler Directive $DebugInfo Determines whether application debug information is built
23321: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
23322: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
23323: Function SysUtils Date Gives the current date
23324: Variable SysUtils DateSeparator The character used to separate display date fields
23325: Function SysUtils DateTimeToFileDate Convert a TDateTime value to a file date/time format
23326: Function SysUtils DateTimeToStr Converts TDateTime date and time values to a string
23327: Procedure SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
23328: Function SysUtils DateToStr Converts a TDateTime date value to a string
23329: Function DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
23330: Function DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
23331: Function DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
23332: Function SysUtils DayOfWeek Gives day of week index for a TDateTime value
23333: Function DateUtils DaysBetween Gives the whole number of days between 2 dates
23334: Function DateUtils DaysInAMonth Gives the number of days in a month
23335: Function DateUtils DaysInAYear Gives the number of days in a year
23336: Function DateUtils DaySpan Gives the fractional number of days between 2 dates
23337: Procedure System Dec Decrement an ordinal variable
23338: Variable SysUtils DecimalSeparator The character used to display the decimal point
23339: Procedure SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
23340: Procedure DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
23341: Procedure SysUtils DecodeTime Break a TDateTime value into individual time values
23342: Directive Default Defines default processing for a property
23343: Function Math DegToRad Convert a degrees value to radians
23344: Procedure System Delete Delete a section of characters from a string
23345: Function SysUtils DeleteFile Delete a file specified by its file name
23346: Keyword Destructor Defines the method used to destroy an object
23347: Function SysUtils DirectoryExists Returns true if the given directory exists
23348: Function SysUtils DiskFree Gives the number of free bytes on a specified drive
23349: Function SysUtils DiskSize Gives the size in bytes of a specified drive
23350: Procedure System Dispose Dispose of storage used by a pointer type variable
23351: Keyword Div Performs integer division, discarding the remainder
23352: Keyword Do Defines the start of some controlled action
23353: Type System Double A floating point type supporting about 15 digits of precision
23354: Keyword DownTo Prefixes an decremental for loop target value
23355: Function StrUtils DupeString Creates a string containing copies of a substring
23356: Directive Dynamic Allows a class method to be overridden in derived classes
23357:
23358: E
23359: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
23360: Compiler Directive $EndIf Terminates conditional code compilation
23361: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
23362: Keyword Else Starts false section of if, case and try statements
23363: Function SysUtils EncodeDate Build a TDateTime value from year, month and day values
23364: Function DateUtils EncodeDateTime Build a TDateTime value from day and time values
23365: Function SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
23366: Keyword End Keyword that terminates statement blocks
23367: Function DateUtils EndOfDayGenerate a TDateTime value set to the very end of a day

```

```

23368: Function DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
23369: Procedure System EndThread Terminates a thread with an exit code
23370: Function System Eof Returns true if a file opened with Reset is at the end
23371: Function System Eoln Returns true if the current text file is pointing at a line end
23372: Procedure System Erase Erase a file
23373: Variable System ErrorAddr Sets the error address when an application terminates
23374: Keyword Except Starts the error trapping clause of a Try statement
23375: Procedure System Exclude Exclude a value in a set variable
23376: Procedure System Exit Exit abruptly from a function or procedure
23377: Variable System ExitCode Sets the return code when an application terminates
23378: Function System Exp Gives the exponent of a number
23379: Directive System Export Makes a function or procedure in a DLL externally available
23380: Type System Extended The floating point type with the highest capacity and precision
23381: Function SysUtils ExtractFileDir Extracts the dir part of a full file name
23382: Function SysUtils ExtractFileDrive Extracts the drive part of a full file name
23383: Function SysUtils ExtractFileExt Extracts the extension part of a full file name
23384: Function SysUtils ExtractFileName Extracts the name part of a full file name
23385: Function SysUtils ExtractFilePath Extracts the path part of a full file name
23386:
23387: F
23388: Function StdConv FahrenheitToCelsius Convert a fahrenheit temperature into celsius
23389: Keyword File Defines a typed or untyped file
23390: Function SysUtils FileAge Get the last modified date/time of a file without opening it
23391: Function SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
23392: Function SysUtils FileExists Returns true if the given file exists
23393: Function SysUtils FileGetAttr Gets the attributes of a file
23394: Variable System FileMode Defines how Reset opens a binary file
23395: Function System FilePos Gives the file position in a binary or text file
23396: Function SysUtils FileSearch Search for a file in one or more directories
23397: Function SysUtils FileSetAttr Sets the attributes of a file
23398: Function SysUtils FileSetDate Set the last modified date and time of a file
23399: Function System FileSize Gives the size in records of an open file
23400: Procedure System FillChar Fills out a section of storage with a fill character or byte value
23401: Keyword Finally Starts the unconditional code section of a Try statement
23402: Function SysUtils FindClose Closes a successful FindFirst file search
23403: Function SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
23404: Function SysUtils FindFirst Finds all files matching a file mask and attributes
23405: Function SysUtils FindNext Find the next file after a successful FindFirst
23406: Function SysUtils FloatToStr Convert a floating point value to a string
23407: Function SysUtils FloatToStrF Convert a floating point value to a string with formatting
23408: Procedure System Flush Flushes buffered text file data to the file
23409: Keyword For Starts a loop that executes a finite number of times
23410: Function SysUtils ForceDirectories Create a new path of directories
23411: Function SysUtils Format Rich formatting of numbers and text into a string
23412: Function SysUtils FormatCurr Rich formatting of a currency value into a string
23413: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
23414: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
23415: Function System Frac The fractional part of a floating point number
23416: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil
23417: Procedure System FreeMem Free memory storage used by a variable
23418: Keyword System Function Defines a subroutine that returns a value
23419:
23420: G
23421: Function SysUtils GetCurrentDir Get the current directory (drive plus directory)
23422: Procedure System GetDir Get the default directory (drive plus path) for a specified drive
23423: Function System GetLastError Gives the error code of the last failing Windows API call
23424: Procedure SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
23425: Function System GetMem Get a specified number of storage bytes
23426: Keyword Goto Forces a jump to a label, regardless of nesting
23427:
23428: H
23429: Compiler Directive $H Treat string types as AnsiString or ShortString
23430: Compiler Directive $Hints Determines whether Delphi shows compilation hints
23431: Procedure System Halt Terminates the program with an optional dialog
23432: Function System Hi Returns the hi-order byte of a (2 byte) Integer
23433: Function System High Returns the highest value of a type or variable
23434:
23435: I
23436: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
23437: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
23438: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
23439: Compiler Directive $IfOpt Tests for the state of a Compiler directive
23440: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
23441: Compiler Directive $IOChecks When on, an IO operation error throws an exception
23442: Keyword If Starts a conditional expression to determine what to do next
23443: Keyword Implementation Starts the implementation (code) section of a Unit
23444: Keyword In Used to test if a value is a member of a set
23445: Procedure System Inc Increment an ordinal variable
23446: Function DateUtils IncDay Increments a TDateTime variable by + or - number of days
23447: Procedure System Include Include a value in a set variable
23448: Function DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
23449: Function DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
23450: Function SysUtils IncMonth Increments a TDateTime variable by a number of months
23451: Function DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
23452: Function DateUtils IncYear Increments a TDateTime variable by a number of years
23453: Directive Index Principally defines indexed class data properties
23454: Constant Math Infinity Floating point value of infinite size
23455: Keyword Inherited Used to call the parent class constructor or destructor method
23456: Variable System Input Defines the standard input text file

```

```

23457: Function Dialogs InputBox Display a dialog that asks for user text input, with default
23458: Function Dialogs InputQuery Display a dialog that asks for user text input
23459: Procedure System Insert Insert a string into another string
23460: Function System Int The integer part of a floating point number as a float
23461: Type System Int64 A 64 bit sized integer - the largest in Delphi
23462: Type System Integer The basic Integer type
23463: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
23464: Function SysUtils IntToHex Convert an Integer into a hexadecimal string
23465: Function SysUtils IntToStr Convert an integer into a string
23466: Function System IOResult Holds the return code of the last I/O operation
23467: Keyword Is Tests whether an object is a certain class or descendant
23468: Function Math IsInfinite Checks whether a floating point number is infinite
23469: Function SysUtils IsLeapYear Returns true if a given calendar year is a leap year
23470: Function System IsMultiThread Returns true if the code is running multiple threads
23471: Function Math IsNaN Checks to see if a floating point number holds a real number
23472:
23473: L
23474: Compiler Directive $L Determines what application debug information is built
23475: Compiler Directive $LocalSymbols Determines what application debug information is built
23476: Compiler Directive $LongStrings Treat string types as AnsiString or ShortString
23477: Function SysUtils LastDelimiter Find the last position of selected characters in a string
23478: Function System Length Return the number of elements in an array or string
23479: Function System Ln Gives the natural logarithm of a number
23480: Function System Lo Returns the low-order byte of a (2 byte) Integer
23481: Function Math Log10 Gives the log to base 10 of a number
23482: Variable SysUtils LongDateFormat Long version of the date to string format
23483: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
23484: Type System LongInt An Integer whose size is guaranteed to be 32 bits
23485: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
23486: Variable SysUtils LongTimeFormat Long version of the time to string format
23487: Type System LongWord A 32 bit unsigned integer
23488: Function System Low Returns the lowest value of a type or variable
23489: Function SysUtils LowerCase Change upper case characters in a string to lower case
23490:
23491: M
23492: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
23493: Function Math Max Gives the maximum of two integer values
23494: Constant System MaxInt The maximum value an Integer can have
23495: Constant System MaxLongInt The maximum value an LongInt can have
23496: Function Math Mean Gives the average for a set of numbers
23497: Function Dialogs MessageDlg Displays a message, symbol, and selectable buttons
23498: Function Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
23499: Function Math Min Gives the minimum of two integer values
23500: Constant SysUtils MinsPerDay Gives the number of minutes in a day
23501: Procedure System MkDir Make a directory
23502: Keyword Mod Performs integer division, returning the remainder
23503: Constant SysUtils Monthdays Gives the number of days in a month
23504: Function DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
23505: Procedure System Move Copy bytes of data from a source to a destination
23506:
23507: N
23508: Constant Math NaN Not a real number
23509: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
23510: Procedure System New Create a new pointer type variable
23511: Constant System Nil A pointer value that is defined as undetermined
23512: Keyword Not Boolean Not or bitwise not of one arguments
23513: Function SysUtils Now Gives the current date and time
23514: Variable Variants Null A variable that has no value
23515:
23516: O
23517: Compiler Directive $O Determines whether Delphi optimises code when compiling
23518: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
23519: Compiler Directive $OverFlowChecks Determines whether Delphi checks integer and enum bounds
23520: Keyword System Object Allows a subroutine data type to refer to an object method
23521: Function System Odd Tests whether an integer has an odd value
23522: Keyword Of Linking keyword used in many places
23523: Keyword On Defines exception handling in a Try Except clause
23524: Keyword Or Boolean or or bitwise or of two arguments
23525: Function System Ord Provides the Ordinal value of an integer, character or enum
23526: Directive Out Identifies a routine parameter for output only
23527: Variable System Output Defines the standard output text file
23528: Directive Overload Allows 2 or more routines to have the same name
23529: Directive Override Defines a method that replaces a virtual parent class method
23530:
23531: P
23532: Keyword Packed Compacts complex data types into minimal storage
23533: Type System PAnsiChar A pointer to an AnsiChar value
23534: Type System PAnsiString Pointer to an AnsiString value
23535: Function System ParamCount Gives the number of parameters passed to the current program
23536: Function System ParamStr Returns one of the parameters used to run the current program
23537: Type System PChar A pointer to an Char value
23538: Type System PCurrency Pointer to a Currency value
23539: Type System PDateTime Pointer to a TDateTime value
23540: Type System PExtended Pointer to a Extended floating point value
23541: Function System Pi The mathematical constant
23542: Type System PInt64 Pointer to an Int64 value
23543: Function Classes Point Generates a TPoint value from X and Y values
23544: Type System Pointer Defines a general use Pointer to any memory based data
23545: Function Classes PointsEqual Compares two TPoint values for equality

```

23546: **Function** System Pos Find the position of one **string** in another  
 23547: **Function** System Pred Decrement an ordinal variable  
 23548: **Function** Printers Printer Returns a reference to the global Printer **object**  
 23549: Directive **Private** Starts the section of **private** data and methods in a **class**  
 23550: Keyword System **Procedure** Defines a subroutine that does not return a value  
 23551: **Procedure** FileCtrl ProcessPath Split a drive/path/filename **string** into its constituent parts  
 23552: Keyword System **Program** Defines the start of an application  
 23553: **Function** Dialogs PromptForFileName Shows a dialog allowing the user to select a **file**  
 23554: Keyword System **Property** Defines controlled access to **class** fields  
 23555: Directive **Protected** Starts a section of **class** **private** data accessible to sub-classes  
 23556: **Type** System PShortString A pointer to an **ShortString** value  
 23557: **Type** System PString Pointer to a **String** value  
 23558: **Function** Types PtInRect Tests to see if a point lies within a rectangle  
 23559: Directive **Public** Starts an externally accessible section of a **class**  
 23560: Directive **Published** Starts a published externally accessible section of a **class**  
 23561: **Type** System PVariant Pointer to a Variant value  
 23562: **Type** System PWideChar Pointer to a WideChar  
 23563: **Type** System PWideString Pointer to a WideString value  
 23564:  
 23565: Q  
 23566: Compiler Directive \$Q Determines whether Delphi checks integer and enum bounds  
 23567:  
 23568: R  
 23569: Compiler Directive \$R Determines whether Delphi checks **array** bounds  
 23570: Compiler Directive \$RangeChecks Determines whether Delphi checks **array** bounds  
 23571: Compiler Directive \$ReferenceInfo Determines whether symbol reference information is built  
 23572: Compiler Directive \$Resource Defines a resource file to be included in the application linking  
 23573: **Function** Math RadToDeg Converts a radian value to degrees  
 23574: Keyword **Raise Raise** an exception  
 23575: **Function** System Random Generate a random floating point or integer number  
 23576: **Procedure** System Randomize Reposition the Random number generator next value  
 23577: **Function** Math RandomRange Generate a random integer number within a supplied range  
 23578: Variable System RandSeed Reposition the Random number generator next value  
 23579: **Procedure** System Read Read data from a binary or text **file**  
 23580: **Procedure** System ReadLn Read a complete line of data from a text **file**  
 23581: **Type** System Real A floating point **type** supporting about 15 digits of precision  
 23582: **Type** System Real48 The floating point **type** with the highest capacity and precision  
 23583: **Procedure** System ReallocMem Reallocate an existing block of storage  
 23584: **Function** DateUtils RecodeDate Change only the date part of a TDateTime variable  
 23585: **Function** DateUtils RecodeTime Change only the time part of a TDateTime variable  
 23586: Keyword Record A structured data **type** - holding fields of data  
 23587: **Function** Classes Rect Create a TRect value from 2 points or 4 coordinates  
 23588: **Function** SysUtils RemoveDir Remove a directory  
 23589: **Procedure** System Rename Rename a **file**  
 23590: **Function** SysUtils RenameFile Rename a **file** or directory  
 23591: Keyword **Repeat Repeat** statements until a termination condition is met  
 23592: **Procedure** SysUtils ReplaceDate Change only the date part of a TDateTime variable  
 23593: **Procedure** SysUtils ReplaceTime Change only the time part of a TDateTime variable  
 23594: **Procedure** System Reset Open a text **file** for reading, or binary **file** for read/write  
 23595: Variable System Result A variable used to hold the return value from a function  
 23596: **Procedure** System ReWrite Open a text or binary **file** for write access  
 23597: **Procedure** System RmDir Remove a directory  
 23598: **Function** System Round Rounds a floating point number to an integer  
 23599: **Procedure** System RunError Terminates the program with an error dialog  
 23600:  
 23601: S  
 23602: Constant SysUtils SecsPerDay Gives the number of seconds in a day  
 23603: **Procedure** System Seek Move the pointer in a binary **file** to a new record position  
 23604: **Function** System SeekEof Skip to the end of the current line or **file**  
 23605: **Function** System SeekEoln Skip to the end of the current line or **file**  
 23606: **Function** FileCtrl SelectDirectory Display a dialog to allow user selection of a directory  
 23607: Variable System Self Hidden parameter to a method - refers to the containing **object**  
 23608: Keyword Set Defines a set of up to 255 distinct values  
 23609: **Function** SysUtils SetCurrentDir Change the current directory  
 23610: **Procedure** System SetLength Changes the size of a **string**, or the size(s) of an **array**  
 23611: **Procedure** System SetString Copies characters from a buffer into a **string**  
 23612: Keyword Shl Shift an integer value left by a number of bits  
 23613: Variable SysUtils ShortDateFormat Compact version of the date to **string** format  
 23614: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday  
 23615: **Type** System ShortInt An integer **type** supporting values -128 to 127  
 23616: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan  
 23617: **Type** System **ShortString** Defines a **string** of up to 255 characters  
 23618: Variable SysUtils ShortTimeFormat Short version of the time to **string** format  
 23619: **Procedure** Dialogs ShowMessage Display a **string** in a simple dialog with an OK button  
 23620: **Procedure** Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button  
 23621: **Procedure** Dialogs ShowMessagePos Display a **string** in a simple dialog at a given screen position  
 23622: Keyword Shr Shift an integer value right by a number of bits  
 23623: **Function** System Sin The Sine of a number  
 23624: **Type** System Single The smallest capacity and precision floating point **type**  
 23625: **Function** System SizeOf Gives the storage byte size of a **type** or variable  
 23626: **Function** System Slice Creates a slice of an **array** as an Open **Array** parameter  
 23627: **Type** System SmallInt An Integer **type** supporting values from -32768 to 32767  
 23628: **Function** System Sqr Gives the square of a number  
 23629: **Function** System Sqrt Gives the square root of a number  
 23630: **Procedure** System Str Converts an integer or floating point number to a **string**  
 23631: **Type** System **String** A data **type** that holds a **string** of characters  
 23632: **Function** System StringOfChar Creates a **string** with one character repeated many times  
 23633: **Function** SysUtils StringReplace Replace one or more substrings found within a **string**  
 23634: **Function** System StringToWideChar Converts a normal **string** into a WideString 0 terminated buffer

```

23635: Function SysUtils StrScan Searches for a specific character in a constant string
23636: Function SysUtils StrToCurr Convert a number string into a currency value
23637: Function SysUtils StrToDate Converts a date string into a TDateTime value
23638: Function SysUtils StrToDateTime Converts a date+time string into a TDateTime value
23639: Function SysUtils StrToFloat Convert a number string into a floating point value
23640: Function SysUtils StrToInt Convert an integer string into an Integer value
23641: Function SysUtils StrToInt64 Convert an integer string into an Int64 value
23642: Function SysUtils StrToInt64Def Convert a string into an Int64 value with default
23643: Function SysUtils StrToIntDef Convert a string into an Integer value with default
23644: Function SysUtils StrToTime Converts a time string into a TDateTime value
23645: Function StrUtils StuffString Replaces a part of one string with another
23646: Function System Succ Increment an ordinal variable
23647: Function Math Sum Return the sum of an array of floating point values
23648:
23649: T
23650: Function Math Tan The Tangent of a number
23651: Type Classes TBits An object that can hold an infinite number of Boolean values
23652: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
23653: Type ConvUtils TConvType Defines a measurement type as used by Convert
23654: Type System TDateTime Data type holding a date and time value
23655: Type System Text Defines a file as a text file
23656: Type System TextFile Declares a file type for storing lines of text
23657: Type SysUtils TFloatFormat Formats for use in floating point number display functions
23658: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
23659: Keyword Then Part of an if statement - starts the true clause
23660: Variable SysUtils ThousandSeparator The character used to display the thousands separator
23661: Keyword ThreadVar Defines variables that are given separate instances per thread
23662: Function SysUtils Time Gives the current time
23663: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
23664: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
23665: Variable SysUtils TimeSeparator The character used to separate display time fields
23666: Function SysUtils TimeToStr Converts a TDateTime time value to a string
23667: Type Classes TList General purpose container of a list of objects
23668: Keyword To Prefixes an incremental for loop target value
23669: Type System TObject The base class type that is ancestor to all other classes
23670: Function DateUtils Tomorrow Gives the date tomorrow
23671: Type Dialogs TOpenDialog Displays a file selection dialog
23672: Type Types TPoint Holds X and Y integer values
23673: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
23674: Type Types TRect Holds rectangle coordinate values
23675: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
23676: Function SysUtils Trim Removes leading and trailing blanks from a string
23677: Function SysUtils TrimLeft Removes leading blanks from a string
23678: Function SysUtils TrimRight Removes trailing blanks from a string
23679: Function System Trunc The integer part of a floating point number
23680: Procedure System Truncate Truncates a file size - removes all data after the current position
23681: Keyword Try Starts code that has error trapping
23682: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
23683: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
23684: Type Classes TStringList Holds a variable length list of strings
23685: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
23686: Type System TThreadFunc Defines the function to be called by BeginThread
23687: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
23688: Keyword Type Defines a new category of variable or process
23689:
23690:
23691: U
23692: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
23693: Keyword Unit Defines the start of a unit file - a Delphi module
23694: Keyword Until Ends a Repeat control loop
23695: Function System UpCase Convert a Char value to upper case
23696: Function SysUtils UpperCase Change lower case characters in a string to upper case
23697: Keyword Uses Declares a list of Units to be imported
23698:
23699: V
23700: Procedure System Val Converts number strings to integer and floating point values
23701: Keyword Var Starts the definition of a section of data variables
23702: Type System Variant A variable type that can hold changing data types
23703: Function Variants VarType Gives the current type of a Variant variable
23704: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
23705: Directive Virtual Allows a class method to be overridden in derived classes
23706:
23707: W
23708: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
23709: Keyword While Repeat statements whilst a continuation condition is met
23710: Type System WideChar Variable type holding a single International character
23711: Function System WideCharToString Copies a null terminated WideChar string to a normal string
23712: Type System WideString A data type that holds a string of WideChars
23713: Keyword With A means of simplifying references to structured variables
23714: Type System Word An integer type supporting values 0 to 65535
23715: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
23716: Procedure System Write Write data to a binary or text file
23717: Procedure System WriteIn Write a complete line of data to a text file
23718:
23719: X
23720: Compiler Directive $X Controls some Pascal extension handling
23721: Keyword Xor Boolean Xor or bitwise Xor of two arguments
23722:
23723: Y

```

```

23724: Compiler Directive $Y Determines whether application symbol information is built
23725: Function DateUtils Yesterday Gives the date yesterday
23726:
23727: Z
23728: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
23729:
23730: mapX:
23731: if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
23732: writeln('cologne map found');
23733: GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
23734: writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
23735: OpenMapX('church trier');
23736: GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
23737: writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));
23738: >>> //latitude: '50.94133705' longitude: '6.95812076100766'
23739: // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
23740: CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TPos): string;
23741:
23742:
23743:
23744: maxbox Ref:
23745: Signature:
23746: SHA1: maxbox3.exe E951E93E67DE0E815D3F7441842994BAEA06F21A
23747: CRC32: maxbox3.exe 78459738
23748: Ref:
23749: 1. writeln(SHA1(Exepath+'\maxbox3.exe'))
23750: 2. shdig: TSHA1Digest;
23751: shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
23752: for i:= 0 to 19 do write(Bytetohex(shdig[i]));
23753:
23754: 3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
23755:
23756: https://www.virustotal.com/en/file/..../
23757:
23758: MD5 62e4fac2c3bb9c78f5b6443c4c89b128
23759: SHA1 e951e93e67de0e815d3f7441842994baea06f21a
23760: SHA256 1ec32d4375fa9e567ce66a52dc30be9f95c5ce7376ec05c51290c5aba6477437
23761: File name: maxbox3.exe
23762:
23763: Copyright Free Pascal Script
23764: Publisher kleiner kommunikation
23765: Product maxbox
23766: Original name maxbox3_9.exe
23767: Internal name Surprise mX4
23768: File version 3.9.9.101
23769: Description maxbox Delphi VM
23770: Comments reduce to the max
23771:
23772: Target machine Intel 386 or later processors and compatible processors
23773: Compilation timestamp 2014-10-26 20:29:16
23774: Entry Point 0x0118CB40
23775: Number of sections 10
23776:
23777: PE sections
23778: Name Virtual address Virtual size Raw size Entropy MD5
23779:
23780: .text 4096 18348040 18348544 6.60 2450e88360e82c0526e83146ddd4153e
23781: .idata 18345176 48260 48640 6.59 ba667d56623f92742675c2510d4a489e
23782: .data 18403328 263080 263168 5.58 31bd78acf5806fd01b23ca6f4c17516
23783: .bss 18669568 381088 0 0.00 d41d8cd98f00b204e9800998ecf8427e
23784: .idata 19054592 54746 54784 5.53 70a112bd2d8ae8d843f66d30f59b94e9
23785: .edata 19111936 77 512 0.90 8398eb8b69908bb05be2b8ad782b06c3
23786: .tls 19116032 208 0 0.00 d41d8cd98f00b204e9800998ecf8427e
23787: .rdata 19120128 24 512 0.28 d31959411398f2479831a94323c3b9c2
23788: .reloc 19124224 1168820 1168896 6.75 c34bf96cf2b3ac802dd9a13a78695d49
23789: .rsrc 20295680 3467264 3467264 5.27 243811ad2eda0722970a7998f2e3818e
23790:
23791: authentihash e18a6419a7849bf13751439fe4b2f5d81e291bc243c014fd04e96cc3f51caf7e
23792: imphash a0d192a8ed78c25f548447101658dad5
23793: File size 22.3 MB ( 23353344 bytes )
23794: File type Win32 EXE
23795: Magic literal
23796: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
23797:
23798: TrID Windows ActiveX control (36.4%)
23799: Inno Setup installer (34.3%)
23800: InstallShield setup (13.4%)
23801: Win32 EXE PECompact compressed (generic) (13.0%)
23802: Win32 Executable (generic) (1.4%)
23803:
23804: RT_BITMAP 743
23805: RT_STRING 304
23806: RT_RCDATA 156
23807: RT_ICON 49
23808: RT_GROUP_ICON 43
23809: RT_CURSOR 31
23810: RT_GROUP_CURSOR 26
23811: WAVE 9
23812: RT_DIALOG 2

```

```
23813: RT_HTML 2
23814: RT_MESSAGETABLE 1
23815: RT_MANIFEST 1
23816: RT_VERSION 1
23817:
23818: ExifTool file metadata
23819: -----
23820: SpecialBuild
23821: mX4
23822: CodeSize
23823: 18397184
23824: SubsystemVersion 4.0
23825: Comments reduce to the max
23826: LinkerVersion 2.25
23827: ImageVersion 0.0
23828: FileSubtype 0
23829: FileVersionNumber 3.9.9.101
23830: LanguageCode German (Swiss)
23831: FileFlagsMask 0x003f
23832: FileDescription maxbox Delphi VM
23833: CharacterSet Windows, Latin1
23834: InitializedDataSize 4955136
23835: FileOS Win32
23836:TimeStamp 2014:10:26 21:29:16+01:00
23837: MIMETYPE application/octet-stream
23838: LegalCopyright Free Pascal Script
23839: FileVersion 3.9.9.101
23840: SpeziellesBuild mX4 Compiler Engine
23841: FileType Win32 EXE
23842: PEtype PE32
23843: InternalName Surprise mX4
23844: FileAccessDate 2014:10:26 23:37:33+01:00
23845: ProductVersion 3.9 Solar mX4
23846: UninitializedDataSize 0
23847: OSVersion 4.0
23848: FileCreateDate 2014:10:26 23:37:33+01:00
23849: LegalTrademarks maxbox
23850: ProductName maxbox
23851: ProductVersionNumber 3.9.9.101
23852: EntryPoint 0x118cb40
23853: ObjectFileType Executable application
23854:
23855: PE imports
23856: [+] AVICAP32.DLL [+] AVICAP32.dll [+] GLU32.dll
23857: [+] IMAGEHLP.DLL [+] MSVCRT.DLL [+] MSVFW32.DLL
23858: [+] OpenGL32.dll [+] SHFolder.dll [+] URLMON.DLL
23859: [+] advapi32.dll [+] comct132.dll [+] comdlg32.dll
23860: [+] gdi32.dll [+] imagehlp.dll [+] imm32.dll
23861: [+] iphlpapi.dll [+] kernel32.dll [+]mpr.dll
23862: [+] msacm32.dll [+] ole32.dll [+] oleacc.dll
23863: [+] oleaut32.dll [+] oledlg.dll [+] opengl32.dll
23864: [+] shell32.dll [+] shlwapi.dll [+] user32.dll
23865: [+] usp10.dll [+] version.dll [+] winhttp.dll
23866: [+] wininet.dll [+] winmm.dll [+] winspool.drv
23867: [+] ws2_32.dll [+] wssock32.dll
23868:
23869: https://www.virustotal.
    com/en/file/1ec32d4375fa9e567ce66a52dc30be9f95c5ce7376ec05c51290c5aba6477437/analysis/1414363050/
23870: ---- bigbitbox code_cleared_checked----
```