

```

1:  *****
2:  Constructor Function and Procedure List of maXbox 3.9.9
3:  *****
4:
5:  //////////////////////////////////////
6:  ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX
7:  -----
8:
9:  File Size of EXE:20079616 V3.9.9.94 April 2014 To EKON/BASTA/JAX/IBZ/SWS
10: *****Now the Funclist*****
11: Funclist Function : 12264 //10766 //10165 (7648)
12: *****Now the Proclist*****
13: Proclist Procedure Size is: 7632 //6792 //6401 4752
14: *****Now Constructors*****
15: Constructlist Constructor Size is: 1253 //995 //
16: def head:max: maXbox7: 12.04.2014 19:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: -----
20: Funclist total Size all is: 21149! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 20161
22: ASize of EXE: 20079616 (16586240) (13511680) (13023744)
23: SHA1 Hash of maXbox 3.9.9.94: 746418954E790A1713A93FF786ABCB50097FCB59
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 ABNFTToText(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 : TINDEDEF
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 AddParamsSQLForDetail(Params:TParams;SQL:WideStr;Native:Bool;QuoteChar:WideString):WideString
105: Function Addr64ToAddr32(const Value: TJclAddr64): TJclAddr32;
106: Function Addr32ToAddr64(const Value: TJclAddr32): TJclAddr64;
107: function AdjustLineBreaksS(const S: string): string
108: TTextLineBreakStyle', '(tlbsLF, tlbsCRLF)
109: Function AdjustLineBreaks(const S: string; Style: TTextLineBreakStyle): string;
110: Function AllData : string
111: function AllocMemCount: integer;
112: function AllocMemSize: integer;
113: Function AllocPatternBitmap( BkColor, FgColor : TColor) : TBitmap
114: Function AllowRegKeyForEveryone( Key : HKEY; Path : string) : Boolean
115: Function AlphaComponent( const Color32 : TColor32) : Integer
116: Function AlphaSort : Boolean
117: Function AlphaSort( ARecurse : Boolean) : Boolean
118: Function AnsiCat( const x, y : AnsiString) : AnsiString
119: Function AnsiCompareFileName( S1, S2 : string) : Integer
120: function AnsiCompareFileName(const S1: string; const S2: string): Integer)
121: Function AnsiCompareStr( S1, S2 : string) : Integer
122: function AnsiCompareStr(const S1: string; const S2: string): Integer;)
123: Function AnsiCompareText( S1, S2 : string) : Integer
124: function AnsiCompareText(const S1: string; const S2: string): Integer;)
125: Function AnsiContainsStr( const AText, ASubText : string) : Boolean
126: Function AnsiContainsText( const AText, ASubText : string) : Boolean
127: Function AnsiCopy( const src : AnsiString; index, count : Integer) : AnsiString
128: Function AnsiDequotedStr( S : string; AQuote : Char) : string
129: Function AnsiEndsStr( const ASubText, AText : string) : Boolean
130: Function AnsiEndsText( const ASubText, AText : string) : Boolean
131: Function AnsiExtractQuotedStr( var Src : PChar; Quote : Char) : string
132: function AnsiExtractQuotedStr(var Src: PChar; Quote: Char): string)
133: Function AnsiIndexStr( const AText : string; const AValues : array of string) : Integer
134: Function AnsiIndexText( const AText : string; const AValues : array of string) : Integer
135: Function AnsiLastChar( S : string) : PChar
136: function AnsiLastChar(const S: string): PChar)
137: Function AnsiLeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString
138: Function AnsiLowerCase( S : string) : string
139: Function AnsiLowercase(s : String) : String;
140: Function AnsiLowerCaseFileName( S : string) : string
141: Function AnsiMatchStr( const AText : string; const AValues : array of string) : Boolean
142: Function AnsiMatchText( const AText : string; const AValues : array of string) : Boolean
143: Function AnsiMidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString
144: Function AnsiPos( const src, sub : AnsiString) : Integer
145: Function AnsiPos( Substr, S : string) : Integer
146: function AnsiPos(const Substr: string; const S: string): Integer;)
147: Function AnsiQuotedStr( S : string; Quote : Char) : string
148: Function AnsiReplaceStr( const AText, AFromText, AToText : string) : string
149: Function AnsiReplaceText( const AText, AFromText, AToText : string) : string
150: Function AnsiResemblesText( const AText, AOther : string) : Boolean
151: Function AnsiReverseString( const AText : AnsiString) : AnsiString
152: Function AnsiRightStr( const AText : AnsiString; const ACount : Integer) : AnsiString
153: function AnsiSameCaption(const Text1: string; const Text2: string): Boolean)
154: Function AnsiSameStr( S1, S2 : string) : Boolean
155: function AnsiSameStr(const S1: string; const S2: string): Boolean)
156: Function AnsiSameText( const S1, S2 : string) : Boolean
157: Function AnsiSameText( S1, S2 : string) : Boolean
158: function AnsiSameText(const S1: string; const S2: string): Boolean)
159: Function AnsiStartsStr( const ASubText, AText : string) : Boolean
160: Function AnsiStartsText( const ASubText, AText : string) : Boolean
161: Function AnsiStrComp( S1, S2 : PChar) : Integer
162: function AnsiStrComp(S1: PChar; S2: PChar): Integer)
163: Function AnsiStrIComp( S1, S2 : PChar) : Integer
164: function AnsiStrIComp(S1: PChar; S2: PChar): Integer)
165: Function AnsiStrLastChar( P : PChar) : PChar
166: function AnsiStrLastChar(P: PChar): PChar)
167: Function AnsiStrLComp( S1, S2 : PChar; MaxLen : Cardinal) : Integer
168: Function AnsiStrLComp( S1, S2 : PChar; MaxLen : Cardinal) : Integer
169: Function AnsiStrLower( Str : PChar) : PChar
170: Function AnsiStrPos( Str, SubStr : PChar) : PChar
171: function AnsiStrPos(Str: PChar; SubStr: PChar): PChar)
172: Function AnsiStrScan(Str: PChar; Chr: Char): PChar)
173: Function AnsiStrUpper( Str : PChar) : PChar
174: Function AnsiToUtf8( const S : string) : UTF8String
175: Function AnsiToUtf8Ex( const S : string; const cp : integer) : UTF8String
176: Function AnsiUpperCase( S : string) : string
177: Function AnsiUppercase(s : String) : String;
178: Function AnsiUpperCaseFileName( S : string) : string

```

```

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

```

```

268: Function BooleanToInteger( const Pb : Boolean) : Integer
269: Function BoolToStr(B: Boolean; UseBoolStrs: Boolean): string
270: Function BoolToStr1(value : boolean) : string;
271: Function booltoint( aBool : Boolean) : LongInt
272: Function inttobool( aInt : LongInt) : Boolean
273: Function Bounds( ALeft, ATop, AWidth, AHeight : Integer) : TRect
274: function Bounds(ALeft, ATop, AWidth, AHeight: Integer): TRect
275: Function BreakApart( BaseString, BreakString : string; StringList : TStringList) : TStringList
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 : TStringList) : Boolean
280: Function Buttons : PTaskDialogButton
281: Function BytesPerScanline( PixelsPerScanline, BitsPerPixel, Alignment : Longint) : Longint
282: Function BytesToCardinal( const AValue : TIdBytes; const AIndex : Integer) : Cardinal
283: Function BytesToChar( const AValue : TIdBytes; const AIndex : Integer) : Char
284: Function BytesToInt64( const AValue : TIdBytes; const AIndex : Integer) : Int64
285: Function BytesToInteger( const AValue : TIdBytes; const AIndex : Integer) : Integer
286: Function BytesToIPv6( const AValue : TIdBytes; const AIndex : Integer) : TIdIPv6Address
287: Function BytesToShort( const AValue : TIdBytes; const AIndex : Integer) : Short
288: Function BytesToString( ABytes: TIdBytes; AStartIndex: Integer; AMaxCount: Integer): string;
289: Function BytesToStr( const Value: TBytes): string;
290: Function BytesToWord( const AValue : TIdBytes; const AIndex : Integer) : Word
291: Function ByteToBin( Int: Byte): string;
292: Function ByteToCharIndex( S : string; Index : Integer) : Integer
293: function ByteToCharIndex(const S: string; Index: Integer): Integer
294: Function ByteToCharLen( S : string; MaxLen : Integer) : Integer
295: function ByteToCharLen(const S: string; MaxLen: Integer): Integer
296: Function ByteToHex( const AByte : Byte) : string
297: Function ByteToOctal( const AByte : Byte) : string
298: Function ByteType( S : string; Index : Integer) : TmbcsByteType
299: function ByteType(const S: string; Index: Integer): TmbcsByteType
300: Function CalcTitleRect( Col : TColumn; ARow : Integer; var MasterCol : TColumn) : TRect
301: Function CalculateDFAFingerprint( oStates : TList) : integer
302: function CallTerminateProcs: Boolean
303: function CANFOCUS:BOOLEAN
304: Function CanLoad( const Ext : string) : Boolean
305: Function CanParse( AWebRequest : TWebRequest) : Boolean
306: Function CanSave( const Ext : string) : Boolean
307: Function CanStart( cChar : char) : boolean
308: Function CaptureScreen : TBitmap;
309: Function CaptureScreen1( Rec : TRect) : TBitmap;
310: Function CardinalToFourChar( ACardinal : LongWord) : string
311: Function CastSoapToNative(Info: PTypeInfo; const SoapData: WideString; NatData: Pointer; IsNull: Boolean): Boolean
312: Function CastSoapToVariant1( SoapInfo : PTypeInfo; const SoapData : WideString) : Variant;
313: Function Ceil( const X : Extended) : Integer
314: Function Ceil16( X : Integer) : Integer
315: Function Ceil4( X : Integer) : Integer
316: Function Ceil8( X : Integer) : Integer
317: Function Ceiling( const X : Extended) : Integer
318: Function CellRect( ACol, ARow : Longint) : TRect
319: Function CelsiusToFahrenheit( const AValue : Double) : Double
320: Function CenterPoint( const Rect : TRect) : TPoint
321: function CenterPoint(const Rect: TRect): TPoint
322: Function ChangeFileExt( FileName, Extension : string) : string
323: function ChangeFileExt(const FileName: string; const Extension: string): string
324: Function CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) : Boolean
325: Function CharInSet( const Ch : Char; const testSet : TSysCharSet) : Boolean
326: Function CharIsInEOF( const AString : string; ACharPos : Integer) : Boolean
327: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : string) : Boolean
328: Function CharLength( S : string; Index : Integer) : Integer
329: Function CharRange( const AMin, AMax : Char) : string
330: function CharSetToIdent(Charset: Longint; var Ident: string): Boolean
331: Function CharToBin(vChr: Char): string;
332: Function CharNext(lpSz: PChar): PChar; stdcall;
333: Function CharToByteIndex( S : string; Index : Integer) : Integer
334: function CharToByteIndex(const S: string; Index: Integer): Integer
335: Function CharToByteLen( S : string; MaxLen : Integer) : Integer
336: function CharToByteLen(const S: string; MaxLen: Integer): Integer
337: Function CharToHex(const APrefix : string; const cc : Char) : shortstring;
338: function CharToHexStr(Value: char): string;
339: function CharToOem(ins, outs: PChar):boolean;
340: function CharToUnicode(Value: Char): string;
341: Function CheckMenuDropdown : Boolean
342: Function CheckMessages : longint
343: Function CheckOpen( Status : DBIResult) : Boolean
344: Function CheckPassword( const APassword : string) : Boolean
345: Function CheckResponse(const AResponse: SmallInt; const AAllowedResponses: array of SmallInt): SmallInt
346: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
347: function CheckSynchronize(Timeout: Integer): Boolean
348: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
349: function ChrA(const a: byte): char;
350: Function ClassIDToProgID(const ClassID: TGUID): string;
351: Function ClassNameIs(const Name: string): Boolean
352: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
353: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
354: Function ClearBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
355: Function ClearBit3( const Value : Word; const Bit : TBitRange) : Word;
356: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;

```



```

357: Function ClearBit5( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
358: Function ClearBit6( const Value : Int64; const Bit : TBitRange) : Int64;
359: function CLIENTTOSCREEN(POINT:TPOINT):TPOINT
360: Function Clipboard : TClipboard
361: Function ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
362: Function ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
363: Function ClipLine( var X1, Y1, X2, Y2 : Integer; const ClipRect : TRect) : Boolean;
364: Function ClipLineToRect( var P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean
365: Function Clone( out stm : IStream) : HRESULT
366: Function CloneConnection : TSQLConnection
367: Function CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
368: function CLOSEQUERY:BOOLEAN
369: Function CloseVolume( var Volume : THandle) : Boolean
370: Function CloseHandle(Handle: Integer): Integer; stdcall;
371: Function CPLApplet( hwndCP1 : THandle; uMsg : DWORD; lParam1, lParam2 : Longint) : Longint
372: Function CmdLine: PChar;
373: function CmdShow: Integer;
374: Function Color32( const R, G, B : Byte; const A : Byte) : TColor32;
375: Function Color32( WinColor : TColor) : TColor32;
376: Function Color321( const Index : Byte; const Palette : TPalette32) : TColor32;
377: Function ColorAdjustLuma( clrRGB : TColor; n : Integer; fScale : Boolean) : TColor
378: Function ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
379: Function ColorToHTML( const Color : TColor) : String
380: function ColorToIdent(Color: Longint; var Ident: string): Boolean)
381: Function ColorToRGB(color: TColor): Longint
382: function ColorToString(Color: TColor): string)
383: Function ColorToWebColorName( Color : TColor) : string
384: Function ColorToWebColorStr( Color : TColor) : string
385: Function ColumnAtDepth( Col : TColumn; ADepth : Integer) : TColumn
386: Function Combination(npr, ncr: integer): extended;
387: Function CombinationInt(npr, ncr: integer): Int64;
388: Function CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo
389: Function CommaAdd( const AStr1, AStr2 : String) : string
390: Function CommercialRound( const X : Extended) : Int64
391: Function Commit( grfCommitFlags : Longint) : HRESULT
392: Function Compare( const NameExt : string) : Boolean
393: function CompareDate(const A, B: TDateTime): TValueRelationship;
394: Function CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Integer
395: Function CompareFiles(const FN1,FN2 :string; Breathe:TNotifyEvent;BreathingSender:TObject): boolean
396: Function CompareMemoryStreams( S1, S2 : TMemoryStream) : boolean
397: Function CompareStr( S1, S2 : string) : Integer
398: function CompareStr(const S1: string; const S2: string): Integer)
399: function CompareString(const S1: string; const S2: string): Integer)
400: Function CompareText( S1, S2 : string) : Integer
401: function CompareText(const S1: string; const S2: string): Integer)
402: Function CompareTextLike(cWildStr,cStr:string; const cWildChar:char;lCaseSensitive:boolean): boolean
403: function CompareTime(const A, B: TDateTime): TValueRelationship;
404: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean
405: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean
406: Function ComponentTypeToString( const ComponentType : DWORD) : string
407: Function CompToCurrency( Value : Comp) : Currency
408: Function CompToDouble( Value : Comp) : Double
409: function ComputeFileCRC32(const FileName : String) : Integer;
410: function ComputeSHA256(astr: string; amode: char): string) //mode F:File, S:String
411: function ComputeSHA512(astr: string; amode: char): string) //mode F:File, S:String
412: Function Concat(s: string): string
413: Function ConnectAndGetAll : string
414: Function Connected : Boolean
415: function constrain(x, a, b: integer): integer;
416: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult
417: Function ConstraintsDisabled : Boolean
418: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
419: Function ContainsState( oState : TniRegularExpressionState) : boolean
420: Function ContainsStr( const AText, ASubText : string) : Boolean
421: Function ContainsText( const AText, ASubText : string) : Boolean
422: Function ContainsTransition( oTransition : TniRegularExpressionTransition) : boolean
423: Function Content : string
424: Function ContentFromStream( Stream : TStream) : string
425: Function ContentFromString( const S : string) : string
426: Function CONTROLSDISABLED : BOOLEAN
427: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
428: Function Convert1( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
429: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
430: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
431: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
432: Function ConvertWriteStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
433: Function ConvFamilyToDescription( const AFamily : TConvFamily) : string
434: Function ConvTypeToDescription( const AType : TConvType) : string
435: Function ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
436: Function ConvTypeToFamily( const AType : TConvType) : TConvFamily;
437: Function ConvAdd(const AVal:Double;const AType1:TConvType;const AVal2:Double;const AType2,
AResultType:TConvType): Double
438: Function ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
AType2:TConvType): TValueRelationship
439: Function ConvDec( const AValue : Double; const AType, AAmountType : TConvType) : Double;
440: Function ConvDec1(const AValue:Double; const AType: TConvType;const AAmount:Double;const
AAmountType:TConvType):Double;
441: Function ConvDiff(const AVal1:Double;const AType1:TConvType;const AVal2:Double;const AType2,
AResuType:TConvType):Double

```

```

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

```

```

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

```

```

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

```



```

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

```

```

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

```

```

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

```

```

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

```



```

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

```

```

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

```

```

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

```

```

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

```



```

1399: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1400: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1401: Function HashValue16( const ASrc : string) : Word;
1402: Function HashValue16Stream( AStream : TStream) : Word;
1403: Function HashValue32( const ASrc : string) : LongWord;
1404: Function HashValue32Stream( AStream : TStream) : LongWord;
1405: Function HasMergeConflicts : Boolean
1406: Function hasMoreTokens : boolean
1407: Function HASPARENT : BOOLEAN
1408: function HasParent: Boolean
1409: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1410: Function HasUTF8BOM( S : TStream) : boolean;
1411: Function HasUTF8BOM1( S : AnsiString) : boolean;
1412: Function Haversine( X : Float) : Float
1413: Function Head( s : string; const subs : string; var tail : string) : string
1414: Function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1415: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1416: function HELPJUMP(JUMPID:STRING):BOOLEAN
1417: Function HeronianMean( const a, b : Float) : Float
1418: function HexStrToStr(Value: string): string;
1419: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1420: function HexToBin2(HexNum: string): string;
1421: Function HexToDouble( const Hex : string) : Double
1422: function HexToInt(hexnum: string): LongInt;
1423: function HexToStr(Value: string): string;
1424: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1425: function Hi(vdat: word): byte;
1426: function HiByte(W: Word): Byte)
1427: function High: Int64;
1428: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1429: function HINSTANCE: longword;
1430: function HiWord(l: DWORD): Word)
1431: function HMODULE: longword;
1432: Function HourOf( const AValue : TDateTime) : Word
1433: Function HourOfDay( const AValue : TDateTime) : Word
1434: Function HourOfTheMonth( const AValue : TDateTime) : Word
1435: Function HourOfTheWeek( const AValue : TDateTime) : Word
1436: Function HourOfTheYear( const AValue : TDateTime) : Word
1437: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1438: Function HourSpan( const ANow, AThen : TDateTime) : Double
1439: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1440: Function HTMLDecode( const AStr : String) : String
1441: Function HTMLEncode( const AStr : String) : String
1442: Function HTMLEscape( const Str : string) : string
1443: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1444: Function HTTPDecode( const AStr : String) : string
1445: Function HTTPEncode( const AStr : String) : string
1446: Function Hypot( const X, Y : Extended) : Extended
1447: Function IBMax( n1, n2 : Integer) : Integer
1448: Function IBMin( n1, n2 : Integer) : Integer
1449: Function IBRandomString( iLength : Integer) : String
1450: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1451: Function IBStripString( st : String; CharsToStrip : String) : String
1452: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1453: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1454: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1455: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1456: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1457: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1458: Function RandomString( iLength : Integer) : String'';
1459: Function RandomInteger( iLow, iHigh : Integer) : Integer'';
1460: Function StripString( st : String; CharsToStrip : String) : String'';
1461: Function FormatIdentifier( Dialect : Integer; Value : String) : String'';
1462: Function FormatIdentifierValue( Dialect : Integer; Value : String) : String'';
1463: Function ExtractIdentifier( Dialect : Integer; Value : String) : String'';
1464: Function QuoteIdentifier( Dialect : Integer; Value : String) : String'';
1465: Function AddIBParamSQLForDetail( Params : TParams; SQL : string; Native : Boolean; Dialect : Integer) :
string'';
1466: Procedure DecomposeDatabaseName( DatabaseName : String; var ServerName, Protocol, DatabasePath :
String'';
1467: function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;'';
1468: Function IconToBitmap( Ico : HICON) : TBitmap
1469: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1470: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1471: function IdentToCharset(const Ident: string; var CharSet: Longint): Boolean)
1472: function IdentToColor(const Ident: string; var Color: Longint): Boolean)
1473: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1474: Function IdGetDefaultCharSet : TIdCharSet
1475: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1476: Function IdPorts2 : TStringList
1477: Function IdToMib( const Id : string) : string
1478: Function IdSHA1Hash( apath: string): string;
1479: Function IdHashSHA1( apath: string): string;
1480: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1481: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1482: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer;
1483: Function iif2( ATest : Boolean; const ATrue : string; const AFalse : string) : string;
1484: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFalse : Boolean) : Boolean;
1485: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;

```

```

1486: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer ) : TDateTime
1487: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64 ) : TDateTime
1488: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte ) : Byte;
1489: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint ) : Shortint;
1490: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint ) : Smallint;
1491: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word ) : Word;
1492: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer ) : Integer;
1493: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal ) : Cardinal;
1494: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64 ) : Int64;
1495: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte ) : Byte;
1496: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint ) : Shortint;
1497: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint ) : Smallint;
1498: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word ) : Word;
1499: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer ) : Integer;
1500: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal ) : Cardinal;
1501: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64 ) : Int64;
1502: Function IncludeTrailingBackslash( S : string ) : string
1503: Function IncludeTrailingBackslash(const S : string) : string
1504: Function IncludeTrailingPathDelimiter( const APath : string ) : string
1505: Function IncludeTrailingPathDelimiter( S : string ) : string
1506: Function IncludeTrailingPathDelimiter(const S : string) : string
1507: Function IncludeTrailingSlash( const APath : string ) : string
1508: Function IncMilliSecond( const AValue : TDateTime; const ANumberOfMilliseconds : Int64 ) : TDateTime
1509: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64 ) : TDateTime
1510: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer ) : TDateTime
1511: Function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime)
1512: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64 ) : TDateTime
1513: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer ) : TDateTime
1514: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer ) : TDateTime
1515: Function IndexOf( AClass : TClass ) : Integer
1516: Function IndexOf( AComponent : TComponent ) : Integer
1517: Function IndexOf( AObject : TObject ) : Integer
1518: Function INDEXOF( const ANAME : String ) : INTEGER
1519: Function IndexOf( const DisplayName : string ) : Integer
1520: Function IndexOf( const Item : TBookmarkStr ) : Integer
1521: Function IndexOf( const S : WideString ) : Integer
1522: Function IndexOf( const View : TJclFileMappingView ) : Integer
1523: Function INDEXOF( FIELD : TFIELD ) : INTEGER
1524: Function IndexOf( ID : LCID ) : Integer
1525: Function INDEXOF( ITEM : TMENUITEM ) : INTEGER
1526: Function IndexOf( Value : TListItem ) : Integer
1527: Function IndexOf( Value : TTreeNode ) : Integer
1528: Function IndexOf(const S: string): Integer;
1529: Function IndexOfName( const Name : WideString ) : Integer
1530: Function IndexOfName(Name: string): Integer;
1531: Function IndexOfObject( AObject : TObject ) : Integer
1532: Function IndexofObject(AObject:TObject):Integer
1533: Function IndexOfTabAt( X, Y : Integer ) : Integer
1534: Function IndexStr( const AText : string; const AValues : array of string ) : Integer
1535: Function IndexText( const AText : string; const AValues : array of string ) : Integer
1536: Function IndexOfInteger( AList : TStringList; Value : Variant ) : Integer
1537: Function IndexOfFloat( AList : TStringList; Value : Variant ) : Integer
1538: Function IndexOfDate( AList : TStringList; Value : Variant ) : Integer
1539: Function IndexOfString( AList : TStringList; Value : Variant ) : Integer
1540: Function IndyCompareStr( const A1 : string; const A2 : string ) : Integer
1541: Function IndyGetHostName : string
1542: Function IndyInterlockedDecrement( var I : Integer ) : Integer
1543: Function IndyInterlockedExchange( var A : Integer; B : Integer ) : Integer
1544: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer ) : Integer
1545: Function IndyInterlockedIncrement( var I : Integer ) : Integer
1546: Function IndyLowerCase( const A1 : string ) : string
1547: Function IndyStrToBool( const AString : String ) : Boolean
1548: Function IndyUpperCase( const A1 : string ) : string
1549: Function InitCommonControl( CC : Integer ) : Boolean
1550: Function InitTempPath : string
1551: Function InMainThread : boolean
1552: Function InOpArray( W : WideChar; sets : array of WideChar ) : boolean
1553: Function Input: Text)
1554: Function InputBox( const ACaption, APrompt, ADefault : string ) : string
1555: Function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1556: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1557: Function InputQuery( const ACaption, APrompt : string; var Value : string ) : Boolean
1558: Function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1559: Function InquireSignal( RtlSigNum : Integer ) : TSignalState
1560: Function InRangeR( const A, Min, Max : Double ) : Boolean
1561: Function Insert( Index : Integer ) : TCollectionItem
1562: Function Insert( Index : Integer ) : TComboExItem
1563: Function Insert( Index : Integer ) : THeaderSection
1564: Function Insert( Index : Integer ) : TListItem
1565: Function Insert( Index : Integer ) : TStatusPanel
1566: Function Insert( Index : Integer ) : TWorkArea
1567: Function Insert( Index : LongInt; const Text : string ) : LongInt
1568: Function Insert( Sibling : TTreeNode; const S : string ) : TTreeNode
1569: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM ) : INTEGER
1570: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM ) : INTEGER
1571: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer ) : TTreeNode
1572: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer ) : LongInt
1573: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer ) : TTreeNode
1574: Function Instance : Longint

```

```

1575: function InstanceSize: Longint
1576: function Int(e : Extended) : Extended;
1577: function Int64ToStr(i: Int64): String;
1578: function IntegerToBcd( const AValue : Integer) : TBcd
1579: function Intensity( const Color32 : TColor32) : Integer;
1580: function Intensity( const R, G, B : Single) : Single;
1581: function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1582: function InterestRate(NPeriods:Integer;const Payment,PresentVal,
FutureVal:Extended;PaymentTime:TPaymentTime): Extended
1583: function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1584: function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1585: function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1586: function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1587: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1588: function IntMibToStr( const Value : string) : string
1589: function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1590: function IntToBin( Value : cardinal) : string
1591: function IntToHex( Value : Integer; Digits : Integer) : string;
1592: function IntToHex(a: integer; b: integer): string;
1593: function IntToHex64( Value : Int64; Digits : Integer) : string;
1594: function IntToHex64(Value: Int64; Digits: Integer): string)
1595: function IntTo3Str( Value : Longint; separator: string) : string
1596: function inttobool( aInt : LongInt) : Boolean
1597: function IntToStr(i: Int64): String;
1598: function IntToStr64(Value: Int64): string)
1599: function IOResult: Integer
1600: function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1601: function IsAccel(VK: Word; const Str: string): Boolean
1602: function IsAddressInNetwork( Address : string) : Boolean
1603: function IsAdministrator : Boolean
1604: function IsAlias( const Name : string) : Boolean
1605: function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1606: function IsASCII( const AByte : Byte) : Boolean;
1607: function IsASCIILDH( const AByte : Byte) : Boolean;
1608: function IsAssembly(const FileName: string): Boolean;
1609: function IsBcdNegative( const Bcd : TBcd) : Boolean
1610: function IsBinary(const AChar : Char) : Boolean
1611: function IsConsole: Boolean)
1612: function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1613: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1614: function IsDelphiDesignMode : boolean
1615: function IsDelphiRunning : boolean
1616: function IsDFSState : boolean
1617: function IsDirectory( const FileName : string) : Boolean
1618: function IsDomain( const S : String) : Boolean
1619: function IsDragObject(Sender: TObject): Boolean;
1620: function IsEditing : Boolean
1621: function ISEMPTY : BOOLEAN
1622: function IsEqual( Value : TParameters) : Boolean
1623: function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1624: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1625: function IsFirstNode : Boolean
1626: function IsFloatZero( const X : Float) : Boolean
1627: function IsFormatRegistered( Extension, AppID : string) : Boolean
1628: function IsFormOpen(const FormName: string): Boolean;
1629: function IsFQDN( const S : String) : Boolean
1630: function IsGrayScale : Boolean
1631: function IsHex( AChar : Char) : Boolean;
1632: function IsHexString(const AString: string): Boolean;
1633: function IsHostname( const S : String) : Boolean
1634: function IsInfinite( const AValue : Double) : Boolean
1635: function IsInLeapYear( const AValue : TDateTime) : Boolean
1636: function IsInternet: boolean;
1637: function IsLeadChar( ACh : Char) : Boolean
1638: function IsLeapYear( Year : Word) : Boolean
1639: function IsLeapYear(Year: Word): Boolean)
1640: function IsLibrary: Boolean)
1641: function ISLINE : BOOLEAN
1642: function IsLinkedTo( DataSet : TDataSet) : Boolean
1643: function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1644: function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1645: function IsMatch( const Pattern, Text : string) : Boolean //Grep like RegEx
1646: function IsMainAppWindow( Wnd : HWND) : Boolean
1647: function IsMediaPresentInDrive( Drive : Char) : Boolean
1648: function IsMemoryManagerSet: Boolean)
1649: function IsMultiTableQuery( const SQL : WideString) : Boolean
1650: function IsMultiThread: Boolean)
1651: function IsNumeric( AChar : Char) : Boolean;
1652: function IsNumeric2( const AString : string) : Boolean;
1653: function IsOctal( AChar : Char) : Boolean;
1654: function IsOctalString(const AString: string) : Boolean;
1655: function IsPathDelimiter( S : string; Index : Integer) : Boolean
1656: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1657: function IsPM( const AValue : TDateTime) : Boolean
1658: function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1659: function IsPrimeFactor( const F, N : Cardinal) : Boolean
1660: function IsPrimeRM( N : Cardinal) : Boolean //rabin miller
1661: function IsPrimeTD( N : Cardinal) : Boolean //trial division

```

```

1662: Function IsPrivilegeEnabled( const Privilege : string ) : Boolean
1663: Function ISqrt( const I : Smallint ) : Smallint
1664: Function IsReadOnly(const Filename: string): boolean;
1665: Function IsRectEmpty( const Rect : TRect ) : Boolean
1666: Function IsRectEmpty(const Rect: TRect): Boolean
1667: Function IsRelativePrime( const X, Y : Cardinal ) : Boolean
1668: Function ISRIGHTTOLEFT : BOOLEAN
1669: Function IsRightToLeft: Boolean
1670: Function IsSameDay( const AValue, ABasis : TDateTime ) : Boolean
1671: Function ISSEQUENCED : BOOLEAN
1672: Function IsSystemModule( const Module : HMODULE ) : Boolean
1673: Function IsSystemResourcesMeterPresent : Boolean
1674: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1675: Function IsToday( const AValue : TDateTime ) : Boolean
1676: Function IsToday(const AValue: TDateTime): Boolean;
1677: Function IsTopDomain( const AStr : string ) : Boolean
1678: Function IsUTF8LeadByte( Lead : Char ) : Boolean
1679: Function IsUTF8String( const s : UTF8String ) : Boolean
1680: Function IsUTF8TrailByte( Lead : Char ) : Boolean
1681: Function ISVALIDCHAR( INPUTCHAR : CHAR ) : BOOLEAN
1682: Function IsValidDate( const AYear, AMonth, ADay : Word ) : Boolean
1683: Function IsValidDateDay( const AYear, ADayOfYear : Word ) : Boolean
1684: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word ) : Boolean
1685: Function IsValidDateTime(const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond: Word): Boolean
1686: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word ) : Boolean
1687: Function IsValidIdent( Ident : string ) : Boolean
1688: function IsValidIdent(const Ident: string; AllowDots: Boolean): Boolean
1689: Function IsValidIP( const S : String ) : Boolean
1690: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word ) : Boolean
1691: Function IsValidISBN( const ISBN : AnsiString ) : Boolean
1692: Function IsVariantManagerSet: Boolean; //deprecated;
1693: Function IsVirtualPcGuest : Boolean;
1694: Function IsVmWareGuest : Boolean;
1695: Function IsVCLControl(Handle: HWND): Boolean;
1696: Function IsWhiteString( const AStr : String ) : Boolean
1697: Function IsWindowResponding( Wnd : HWND; Timeout : Integer ) : Boolean
1698: Function IsWoW64: boolean;
1699: Function IsWin64: boolean;
1700: Function IsWow64String(var s: string): Boolean;
1701: Function IsWin64String(var s: string): Boolean;
1702: Function IsWindowsVista: boolean;
1703: Function isPowerof2(num: int64): boolean;
1704: Function powerOf2(exponent: integer): int64;
1705: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1706: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1707: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1708: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean ) : Integer
1709: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1710: Function ItemRect( Index : Integer ) : TRect
1711: function ITEMRECT(INDEX:INTEGER):TRECT
1712: Function ItemWidth( Index : Integer ) : Integer
1713: Function JavahashCode(val: string): Integer;
1714: Function JosephusG(n,k: integer; var graphout: string): integer;
1715: Function JulianDateToDateTime( const AValue : Double ) : TDateTime
1716: Function JvMessageBox( const Text, Caption : string; Flags : DWORD ) : Integer;
1717: Function JvMessageBox1( const Text : string; Flags : DWORD ) : Integer;
1718: Function KeepAlive : Boolean
1719: Function KeysToShiftState(Keys: Word): TShiftState;
1720: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1721: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1722: Function KeyboardStateToShiftState: TShiftState; overload;
1723: Function Languages : TLanguages
1724: Function Last : TClass
1725: Function Last : TComponent
1726: Function Last : TObject
1727: Function LastDelimiter( Delimiters, S : string ) : Integer
1728: Function LastDelimiter(const Delimiters: string; const S: string): Integer
1729: Function LastPos( const ASubstr : string; const AStr : string ) : Integer
1730: Function Latitude2WGS84(lat: double): double;
1731: Function LCM(m,n:longint):longint;
1732: Function LCMJ( const X, Y : Cardinal ) : Cardinal
1733: Function Ldexp( const X : Extended; const P : Integer ) : Extended
1734: Function LeftStr( const AText : AnsiString; const ACount : Integer ) : AnsiString;
1735: Function LeftStr( const AText : WideString; const ACount : Integer ) : WideString;
1736: function Length: Integer;
1737: Procedure LetFileList(FileList: TStringlist; apath: string);
1738: function lengthmp3(mp3path: string):integer;
1739: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect ) : Boolean;
1740: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect ) : Boolean;
1741: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint; L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1742: function LineStart(Buffer, BufPos: PChar): PChar
1743: function LineStart(Buffer, BufPos: PChar): PChar
1744: function ListSeparator: char;
1745: function Ln(x: Extended): Extended;
1746: Function LnXP1( const X : Extended ) : Extended
1747: function Lo(vdat: word): byte;
1748: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1749: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean ) : Boolean

```



```

1750: Function LoadFileAsString( const FileName : string ) : string
1751: Function LoadFromFile( const FileName : string ) : TBitmapLoader
1752: Function LoadLibraryEx( LibName: PChar; hFile: Longint; Flags: Longint ): Longint; stdcall;
1753: Function LoadPackage( const Name: string ) : HMODULE
1754: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle ) : HGLOBAL
1755: Function LoadStr( Ident : Integer ) : string
1756: Function LoadString( Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer ): Integer; stdcall;
1757: Function LoadWideStr( Ident : Integer ) : WideString
1758: Function LOCATE( const KEYFIELDS: string; const KEYVALUES: VARIANT; OPTIONS: TLOCATEOPTIONS ) : BOOLEAN
1759: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint ) : HRESULT
1760: Function LockServer( fLock : LongBool ) : HRESULT
1761: Function LockVolume( const Volume : string; var Handle : THandle ) : Boolean
1762: Function Log( const X : Extended ) : Extended
1763: Function Log10( const X : Extended ) : Extended
1764: Function Log2( const X : Extended ) : Extended
1765: function LogBase10( X: Float ): Float;
1766: Function LogBase2( X: Float ): Float;
1767: Function LogBaseN( Base, X: Float ): Float;
1768: Function LogN( const Base, X : Extended ) : Extended
1769: Function LogOffOS : Boolean
1770: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string ) : Boolean
1771: Function LoginDialogEx( const ADatabaseName: string; var AUserName, APassword: string; NameReadOnly: Bool ): Bool;
1772: Function LongDateFormat: string;
1773: function LongTimeFormat: string;
1774: Function LongWordToFourChar( ACardinal : LongWord ) : string
1775: Function LOOKUP( const KEYFIELDS: string; const KEYVALUES: VARIANT; const RESULTFIELDS: string ): VARIANT
1776: Function LookupName( const name : string ) : TInAddr
1777: Function LookupService( const service : string ) : Integer
1778: function Low: Int64;
1779: Function LowerCase( S : string ) : string
1780: Function Lowercase( s : AnyString ) : AnyString;
1781: Function LRot( const Value : Byte; const Count : TBitRange ) : Byte;
1782: Function LRotl( const Value : Word; const Count : TBitRange ) : Word;
1783: Function LRot2( const Value : Integer; const Count : TBitRange ) : Integer;
1784: function MainInstance: longword
1785: function MainThreadID: longword
1786: Function Map( x, in_min, in_max, out_min, out_max: integer ): integer; //arduino
1787: Function mapMax( ax, in_min, in_max, out_min, out_max: integer ): integer;
1788: Function MakeCanonicalIPv4Address( const AAddr : string ) : string
1789: Function MakeCanonicalIPv6Address( const AAddr : string ) : string
1790: Function MakeDIB( out Bitmap : PBitmapInfo ) : Integer
1791: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal ) : string
1792: function MakeLong( A, B: Word ): Longint
1793: Function MakeTempFilename( const APath : string ) : string
1794: Function MakeValidFileName( const Str : string ) : string
1795: Function MakeValueMap( Enumeration : string; ToCds : Boolean ) : string
1796: function MakeWord( A, B: Byte ): Word
1797: Function MakeYear4Digit( Year, Pivot : Integer ) : Integer
1798: Function MapDateTime( const DateFormatType: string; DateFormat: string; Value: string; ToCds: Boolean ): string
1799: Function MapValues( Mapping : string; Value : string ) : string
1800: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char ) : string
1801: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer ) : TMaskCharType
1802: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer ) : TMaskDirectives
1803: Function MaskGetFldSeparator( const EditMask : string ) : Integer
1804: Function MaskGetMaskBlank( const EditMask : string ) : Char
1805: Function MaskGetMaskSave( const EditMask : string ) : Boolean
1806: Function MaskIntlLiteralToChar( IChar : Char ) : Char
1807: Function MaskOffsetToOffset( const EditMask : string; MaskOffset : Integer ) : Integer
1808: Function MaskOffsetToWideOffset( const EditMask : string; MaskOffset : Integer ) : Integer
1809: Function MaskString( Mask, Value : string ) : string
1810: Function Match( const sString : string ) : TniRegularExpressionMatchResul
1811: Function Matchl( const sString : string; iStart : integer ) : TniRegularExpressionMatchResult
1812: Function Matches( const Filename : string ) : Boolean
1813: Function MatchesMask( const Filename, Mask : string ) : Boolean
1814: Function MatchStr( const AText : string; const AValues : array of string ) : Boolean
1815: Function MatchText( const AText : string; const AValues : array of string ) : Boolean
1816: Function Max( AValueOne, AValueTwo : Integer ) : Integer
1817: function Max( const x, y: Integer ): Integer;
1818: Function Max1( const B1, B2 : Shortint ) : Shortint;
1819: Function Max2( const B1, B2 : Smallint ) : Smallint;
1820: Function Max3( const B1, B2 : Word ) : Word;
1821: function Max3( const x, y, z: Integer ): Integer;
1822: Function Max4( const B1, B2 : Integer ) : Integer;
1823: Function Max5( const B1, B2 : Cardinal ) : Cardinal;
1824: Function Max6( const B1, B2 : Int64 ) : Int64;
1825: Function Max64( const AValueOne, AValueTwo : Int64 ) : Int64
1826: Function MaxFloat( const X, Y : Float ) : Float
1827: Function MaxFloatArray( const B : TDynFloatArray ) : Float
1828: Function MaxFloatArrayIndex( const B : TDynFloatArray ) : Integer
1829: function MaxIntValue( const Data: array of Integer ): Integer
1830: Function MaxJ( const B1, B2 : Byte ) : Byte;
1831: function MaxPath: string;
1832: function MaxValue( const Data: array of Double ): Double
1833: Function MaxCalc( const Formula : string ) : Extended //math expression parser
1834: Procedure MaxCalcF( const Formula : string ); //out to console memo2
1835: function MD5( const fileName: string ): string;
1836: Function Mean( const Data : array of Double ) : Extended
1837: Function Median( const X : TDynFloatArray ) : Float
1838: Function Memory : Pointer

```

```

1839: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer ) : Integer
1840: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1841: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1842: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1843: Function MessageDlg1(const
    Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1844: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
    Y:Integer):Integer;
1845: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
    Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn ) : Integer;
1846: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
    Longint; X, Y : Integer; const HelpFileName : string ) : Integer;
1847: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
    : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn ) : Integer;
1848: Function MibToId( Mib : string ) : string
1849: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer ) : AnsiString;
1850: Function MidStr( const AText : WideString; const AStart, ACount : Integer ) : WideString;
1851: Function microsecondsToCentimeters(mseconds: longint): longint; //340m/s speed of sound
1852: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1853: Function MIDIOut( DeviceID : Cardinal ) : IJclMIDIOut
1854: Procedure GetMidiOutputs( const List : TStrings)
1855: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single ) : TSingleNoteTuningData
1856: Function MIDINoteToStr( Note : TMIDINote ) : string
1857: Function WinMidiOut( DeviceID : Cardinal ) : IJclWinMidiOut
1858: Procedure GetMidiOutputs( const List : TStrings)
1859: Procedure MidiOutCheck( Code : MMResult)
1860: Procedure MidiInCheck( Code : MMResult)
1861: Function MilliSecondOf( const AValue : TDateTime ) : Word
1862: Function MilliSecondOfDay( const AValue : TDateTime ) : LongWord
1863: Function MilliSecondOfTheHour( const AValue : TDateTime ) : LongWord
1864: Function MilliSecondOfTheMinute( const AValue : TDateTime ) : LongWord
1865: Function MilliSecondOfTheMonth( const AValue : TDateTime ) : LongWord
1866: Function MilliSecondOfTheSecond( const AValue : TDateTime ) : Word
1867: Function MilliSecondOfTheWeek( const AValue : TDateTime ) : LongWord
1868: Function MilliSecondOfTheYear( const AValue : TDateTime ) : Int64
1869: Function MilliSecondsBetween( const ANow, AThen : TDateTime ) : Int64
1870: Function MilliSecondSpan( const ANow, AThen : TDateTime ) : Double
1871: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean ) : Int64
1872: Function millis: int64;
1873: Function Min( AValueOne, AValueTwo : Integer ) : Integer
1874: Function Min1( const B1, B2 : Shortint ) : Shortint;
1875: Function Min2( const B1, B2 : Smallint ) : Smallint;
1876: Function Min3( const B1, B2 : Word ) : Word;
1877: Function Min4( const B1, B2 : Integer ) : Integer;
1878: Function Min5( const B1, B2 : Cardinal ) : Cardinal;
1879: Function Min6( const B1, B2 : Int64 ) : Int64;
1880: Function Min64( const AValueOne, AValueTwo : Int64 ) : Int64
1881: Function MinClientRect : TRect;
1882: Function MinClientRect1( IncludeScroller : Boolean ) : TRect;
1883: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean ) : TRect;
1884: Function MinFloat( const X, Y : Float ) : Float
1885: Function MinFloatArray( const B : TDynFloatArray ) : Float
1886: Function MinFloatArrayIndex( const B : TDynFloatArray ) : Integer
1887: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer ) : string
1888: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer ) : TFileName
1889: Function MinimizeName(const Filename: string; Canvas: TCanvas;MaxLen: Integer): TFileName
1890: Function MinIntValue( const Data : array of Integer ) : Integer
1891: function MinIntValue(const Data: array of Integer):Integer)
1892: Function MinJ( const B1, B2 : Byte ) : Byte;
1893: Function MinuteOf( const AValue : TDateTime ) : Word
1894: Function MinuteOfDay( const AValue : TDateTime ) : Word
1895: Function MinuteOfTheHour( const AValue : TDateTime ) : Word
1896: Function MinuteOfTheMonth( const AValue : TDateTime ) : Word
1897: Function MinuteOfTheWeek( const AValue : TDateTime ) : Word
1898: Function MinuteOfTheYear( const AValue : TDateTime ) : LongWord
1899: Function MinutesBetween( const ANow, AThen : TDateTime ) : Int64
1900: Function MinuteSpan( const ANow, AThen : TDateTime ) : Double
1901: Function MinValue( const Data : array of Double ) : Double
1902: function MinValue(const Data: array of Double): Double)
1903: Function MixerLeftRightToArray( Left, Right : Cardinal ) : TDynCardinalArray
1904: Function MMCheck( const MciError : MCIERROR; const Msg : string ) : MCIERROR
1905: Function ModFloat( const X, Y : Float ) : Float
1906: Function ModifiedJulianDateToDateTime( const AValue : Double ) : TDateTime
1907: Function Modify( const Key : string; Value : Integer ) : Boolean
1908: Function ModuleCacheID : Cardinal
1909: Function ModuleFromAddr( const Addr : Pointer ) : HMODULE
1910: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo ) : TMonitor
1911: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo ) : TMonitor
1912: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo ) : TMonitor
1913: Function MonthOf( const AValue : TDateTime ) : Word
1914: Function MonthOfTheYear( const AValue : TDateTime ) : Word
1915: Function MonthsBetween( const ANow, AThen : TDateTime ) : Integer
1916: Function MonthSpan( const ANow, AThen : TDateTime ) : Double
1917: Function MonthStr( DateTime : TDateTime ) : string
1918: Function MouseCoord( X, Y : Integer ) : TGridCoord
1919: Function MOVEBY( DISTANCE : INTEGER ) : INTEGER
1920: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS ) : Boolean
1921: Function MoveNext : Boolean
1922: Function MSecsToTimeStamp( MSecs : Comp ) : TTimeStamp

```

```

1923: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1924: Function Name : string
1925: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
Double;PaymentTime:TPaymentTime):Extended
1926: function NetworkVolume(DriveChar: Char): string
1927: Function NEWBOTOMLINE : INTEGER
1928: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant ) : PExprNode
1929: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
TNOTIFYEVENT; HCTX : WORD; const ANAME : String ) : TMENUITEM
1930: Function NEWLINE : TMENUITEM
1931: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem ) : TMAINMENU
1932: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
Right:PExprNode):PExprNode
1933: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPOPALIGNMENT;AUTOPOPUP:BOOLEAN;
const ITEMS : array of TCMENUITEM ) : TPOPUPMENU
1934: Function NewState( eType : TniRegularExpressionStateType ) : TniRegularExpressionState
1935: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
TMenuItem;AENABLED:BOOL): TMENUITEM
1936: Function NEWTOPLINE : INTEGER
1937: Function Next : TIdAuthWhatsNext
1938: Function NextCharIndex( S : String; Index : Integer ) : Integer
1939: Function NextRecordSet : TCustomSQLDataSet
1940: Function NextRecordset( var RecordsAffected : Integer ) : _Recordset
1941: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken ) : TSQLToken;
1942: Function NextToken : Char
1943: Function nextToken : WideString
1944: function NextToken:Char
1945: Function Norm( const Data : array of Double ) : Extended
1946: Function NormalizeAngle( const Angle : Extended ) : Extended
1947: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word ) : Boolean
1948: Function NormalizeRect( const Rect : TRect ) : TRect
1949: function NormalizeRect(const Rect: TRect): TRect;
1950: Function Now : TDateTime
1951: function Now2: tDateTime
1952: Function NumProcessThreads : integer
1953: Function NumThreadCount : integer
1954: Function NthDayOfWeek( const AValue : TDateTime ) : Word
1955: Function NtProductType : TntProductType
1956: Function NtProductTypeString : string
1957: function Null: Variant;
1958: Function NullPoint : TPoint
1959: Function NullRect : TRect
1960: Function Null2Blank(aString:String):String;
1961: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
Extended; PaymentTime : TPaymentTime ) : Extended
1962: Function NumIP : integer
1963: function Odd(x: Longint): boolean;
1964: Function OffsetFromUTC : TDateTime
1965: Function OffsetPoint( const P, Offset : TPoint ) : TPoint
1966: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer ) : Boolean
1967: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
1968: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer ) : Integer
1969: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment ) : Boolean
1970: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency ) : Boolean
1971: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
1972: function OpenBit:Integer
1973: Function OpenDatabase : TDatabase
1974: Function OpenDatabase( const DatabaseName : string ) : TDatabase
1975: Function OpenGLColorToWinColor( const Red, Green, Blue : Float ) : TColor
1976: Function OpenObject( Value : PChar ) : Boolean;
1977: Function OpenObject1( Value : string ) : Boolean;
1978: Function OpenSession( const SessionName : string ) : TSession
1979: Function OpenVolume( const Drive : Char ) : THandle
1980: function OrdFourByteToCardinal(ABYTE1, AByte2, AByte3, AByte4 : Byte): Cardinal
1981: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte ) : LongWord
1982: Function OrdToBinary( const Value : Byte ) : string;
1983: Function OrdToBinary1( const Value : Shortint ) : string;
1984: Function OrdToBinary2( const Value : Smallint ) : string;
1985: Function OrdToBinary3( const Value : Word ) : string;
1986: Function OrdToBinary4( const Value : Integer ) : string;
1987: Function OrdToBinary5( const Value : Cardinal ) : string;
1988: Function OrdToBinary6( const Value : Int64 ) : string;
1989: Function OSCheck( RetVal : Boolean ) : Boolean
1990: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD ) : string
1991: Function OSIdentToString( const OSIdent : DWORD ) : string
1992: Function Output: Text)
1993: Function Overlay( ImageIndex : Integer; Overlay : TOverlay ) : Boolean
1994: Function Owner : TCustomListView
1995: function Owner : TPersistent
1996: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char ) : string
1997: Function PadL( pStr : String; pLth : integer ) : String
1998: Function Padl(s : AnyString;I : longInt) : AnyString;
1999: Function PadLCh( pStr : String; pLth : integer; pChr : char ) : String
2000: Function PadR( pStr : String; pLth : integer ) : String
2001: Function Padr(s : AnyString;I : longInt) : AnyString;
2002: Function PadRCh( pStr : String; pLth : integer; pChr : char ) : String
2003: Function PadString( const AString : String; const ALen : Integer; const AChar : Char ) : String
2004: Function Padz(s : AnyString;I : longInt) : AnyString;
2005: Function PaethPredictor( a, b, c : Byte ) : Byte

```

```

2006: Function PARAMBYNAME( const VALUE : String ) : TPARAM
2007: Function ParamByName( const Value : WideString ) : TParameter
2008: Function ParamCount: Integer
2009: Function ParamsEncode( const ASrc : string ) : string
2010: function ParamStr(Index: Integer): string
2011: Function ParseDate( const DateStr : string ) : TDateTime
2012: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN ) : String
2013: Function ParseSQL( SQL : WideString; DoCreate : Boolean ) : WideString
2014: Function PathAddExtension( const Path, Extension : string ) : string
2015: Function PathAddSeparator( const Path : string ) : string
2016: Function PathAppend( const Path, Append : string ) : string
2017: Function PathBuildRoot( const Drive : Byte ) : string
2018: Function PathCanonicalize( const Path : string ) : string
2019: Function PathCommonPrefix( const Path1, Path2 : string ) : Integer
2020: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2021: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2022: Function PathEncode( const ASrc : string ) : string
2023: Function PathExtractFileDirFixed( const S : AnsiString ) : AnsiString
2024: Function PathExtractFileNameNoExt( const Path : string ) : string
2025: Function PathExtractPathDepth( const Path : string; Depth : Integer ) : string
2026: Function PathGetDepth( const Path : string ) : Integer
2027: Function PathGetLongName( const Path : string ) : string
2028: Function PathGetLongName2( Path : string ) : string
2029: Function PathGetShortName( const Path : string ) : string
2030: Function PathIsAbsolute( const Path : string ) : Boolean
2031: Function PathIsChild( const Path, Base : AnsiString ) : Boolean
2032: Function PathIsDiskDevice( const Path : string ) : Boolean
2033: Function PathIsUNC( const Path : string ) : Boolean
2034: Function PathRemoveExtension( const Path : string ) : string
2035: Function PathRemoveSeparator( const Path : string ) : string
2036: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2037: Function Peek : Pointer
2038: Function Peek : TObject
2039: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2040: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
Extended; PaymentTime : TPaymentTime ) : Extended
2041: function Permutation(npr, k: integer): extended;
2042: function PermutationInt(npr, k: integer): Int64;
2043: Function PermutationJ( N, R : Cardinal ) : Float
2044: Function Pi : Extended;
2045: Function PiE : Extended;
2046: Function PixelsToDialogsX( const Pixels : Word ) : Word
2047: Function PixelsToDialogsY( const Pixels : Word ) : Word
2048: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2049: Function Point( X, Y : Integer ) : TPoint
2050: function Point(X, Y: Integer): TPoint)
2051: Function PointAssign( const X, Y : Integer ) : TPoint
2052: Function PointDist( const P1, P2 : TPoint ) : Double;
2053: function PointDist(const P1,P2: TFloatPoint): Double;
2054: Function PointDist1( const P1, P2 : TFloatPoint ) : Double;
2055: function PointDist2(const P1,P2: TPoint): Double;
2056: Function PointEqual( const P1, P2 : TPoint ) : Boolean
2057: Function PointIsNull( const P : TPoint ) : Boolean
2058: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint ) : Double
2059: Function Poly( const X : Extended; const Coefficients : array of Double ) : Extended
2060: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2061: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2062: Function Pop : Pointer
2063: Function Pop : TObject
2064: Function PopnStdDev( const Data : array of Double ) : Extended
2065: Function PopnVariance( const Data : array of Double ) : Extended
2066: Function PopulationVariance( const X : TDynFloatArray ) : Float
2067: function Pos(SubStr, S: AnyString): Longint;
2068: Function PosEqual( const Rect : TRect ) : Boolean
2069: Function PosEx( const SubStr, S : string; Offset : Integer ) : Integer
2070: Function PosInSmallIntArray( const ASearchInt: SmallInt; AArray : array of SmallInt ) : Integer
2071: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2072: Function Post1( AURL : string; const ASource : TStringList ) : string;
2073: Function Post2( AURL : string; const ASource : TStream ) : string;
2074: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream ) : string;
2075: Function PostData( const UserData : WideString; const CheckSum : DWORD ) : Boolean
2076: Function PostData( const UserData : WideString; const CheckSum : integer ) : Boolean
2077: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2078: Function Power( const Base, Exponent : Extended ) : Extended
2079: Function PowerBig(aval, n:integer): string;
2080: Function PowerIntJ( const X : Float; N : Integer ) : Float;
2081: Function PowerJ( const Base, Exponent : Float ) : Float;
2082: Function PowerOffOS : Boolean
2083: Function PreformatDateString( Ps : string ) : string
2084: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2085: Function PrimeFactors( N : Cardinal ) : TDynCardinalArray
2086: Function Printer : TPrinter
2087: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string) : string
2088: Function ProcessResponse : TIdHTTPWhatsNext
2089: Function ProduceContent : string
2090: Function ProduceContentFromStream( Stream : TStream ) : string
2091: Function ProduceContentFromString( const S : string ) : string

```



```

2092: Function ProgIDToClassID(const ProgID: string): TGUID;
2093: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2094: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2095: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2096: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string; const
ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2097: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2098: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2099: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2100: Function Push( AItem : Pointer) : Pointer
2101: Function Push( AObject : TObject) : TObject
2102: Function Put1( AURL : string; const ASource : TStream) : string;
2103: Function Pythagoras( const X, Y : Extended) : Extended
2104: Function queryDLLInterface( var queryList : TStringList) : TStringList
2105: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2106: Function QueryInterface(const IID: TGUID; out Obj): HRESULT, CdStdCall
2107: Function queryPerformanceCounter2(mse: int64): int64;
2108: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2109: //Function QueryPerformanceFrequency(mse: int64): boolean;
2110: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2111: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2112: Procedure QueryPerformanceCounter1(var aC: Int64);
2113: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2114: Function Quote( const ACommand : String) : SmallInt
2115: Function QuotedStr( S : string) : string
2116: Function RadToCycle( const Radians : Extended) : Extended
2117: Function RadToDeg( const Radians : Extended) : Extended
2118: Function RadToDeg( const Value : Extended) : Extended;
2119: Function RadToDeg1( const Value : Double) : Double;
2120: Function RadToDeg2( const Value : Single) : Single;
2121: Function RadToGrad( const Radians : Extended) : Extended
2122: Function RadToGrad( const Value : Extended) : Extended;
2123: Function RadToGrad1( const Value : Double) : Double;
2124: Function RadToGrad2( const Value : Single) : Single;
2125: Function RandG( Mean, StdDev : Extended) : Extended
2126: function Random(const ARange: Integer): Integer;
2127: function random2(a: integer): double
2128: function RandomE: Extended;
2129: function RandomF: Extended;
2130: Function RandomFrom( const AValues : array of string) : string;
2131: Function RandomRange( const AFrom, ATo : Integer) : Integer
2132: function randSeed: longint
2133: Function RawToDataColumn( ACol : Integer) : Integer
2134: Function Read : Char
2135: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HRESULT
2136: function Read(Buffer:String;Count:LongInt):LongInt
2137: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2138: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2139: Function ReadCardinal( const AConvert : boolean) : Cardinal
2140: Function ReadChar : Char
2141: Function ReadClient( var Buffer, Count : Integer) : Integer
2142: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2143: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2144: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2145: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:
Boolean):Integer
2146: Function ReadInteger( const AConvert : boolean) : Integer
2147: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2148: Function ReadLn : string
2149: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2150: function ReadLn(question: string): string;
2151: Function ReadLnWait( AFailCount : Integer) : string
2152: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2153: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2154: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2155: Function ReadString( const ABytes : Integer) : string
2156: Function ReadString( const Section, Ident, Default : string) : string
2157: Function ReadString( Count : Integer) : string
2158: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2159: Function ReadTimeStampCounter : Int64
2160: Function RebootOS : Boolean
2161: Function Receive( ATimeout : Integer) : TReplyStatus
2162: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2163: Function ReceiveLength : Integer
2164: Function ReceiveText : string
2165: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2166: Function ReceiveSerialText: string
2167: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2168: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
AMilliSec:Word):TDateTime
2169: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2170: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2171: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2172: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2173: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2174: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2175: Function RecodeTime( const AValue : TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2176: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime

```

```

2177: Function Reconcile( const Results : OleVariant) : Boolean
2178: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2179: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2180: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2181: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2182: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2183: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2184: Function RectCenter( const R : TRect) : TPoint
2185: Function RectEqual( const R1, R2 : TRect) : Boolean
2186: Function RectHeight( const R : TRect) : Integer
2187: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2188: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2189: Function RectIntersection( const R1, R2 : TRect) : TRect
2190: Function RectIntersectRect( const R1, R2 : TRect) : TRect
2191: Function RectIsEmpty( const R : TRect) : Boolean
2192: Function RectIsNull( const R : TRect) : Boolean
2193: Function RectIsSquare( const R : TRect) : Boolean
2194: Function RectIsValid( const R : TRect) : Boolean
2195: Function RectsAreValid( R : array of TRect) : Boolean
2196: Function RectUnion( const R1, R2 : TRect) : TRect
2197: Function RectWidth( const R : TRect) : Integer
2198: Function RedComponent( const Color32 : TColor32) : Integer
2199: Function Refresh : Boolean
2200: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2201: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2202: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2203: Function RegisterConversionType(const AFam:TConvFam;const ADescr:string;const AFact:Double):TConvType
2204: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2205: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2206: Function ReleaseHandle : HBITMAP
2207: Function ReleaseHandle : HENHMETAFILE
2208: Function ReleaseHandle : HICON
2209: Function ReleasePalette : HPALETTE
2210: Function RemainderFloat( const X, Y : Float) : Float
2211: Function Remove( AClass : TClass) : Integer
2212: Function Remove( AComponent : TComponent) : Integer
2213: Function Remove( AItem : Integer) : Integer
2214: Function Remove( AItem : Pointer) : Pointer
2215: Function Remove( Aitem : TObject) : TObject
2216: Function Remove( AObject : TObject) : Integer
2217: Function RemoveBackslash( const PathName : string) : string
2218: Function RemoveDF( aString : String) : String //removes thousand separator
2219: Function RemoveDir( Dir : string) : Boolean
2220: function RemoveDir(const Dir: string): Boolean)
2221: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2222: Function RemoveFileExt( const FileName : string) : string
2223: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2224: Function RenameFile( OldName, NewName : string) : Boolean
2225: function RenameFile(const OldName: string; const NewName: string): Boolean)
2226: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2227: Function ReplaceText( const AText, AFromText, AToText : string) : string
2228: Function Replicate(c : char;I : longInt) : String;
2229: Function Request : TWebRequest
2230: Function ResemblesText( const AText, AOther : string) : Boolean
2231: Function Reset : Boolean
2232: function Reset2(mypath: string):string;
2233: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2234: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2235: Function Response : TWebResponse
2236: Function ResumeSupported : Boolean
2237: Function RETHINKHOTKEYS : BOOLEAN
2238: Function RETHINKLINES : BOOLEAN
2239: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2240: Function RetrieveCurrentDir : string
2241: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2242: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2243: Function RetrieveMailBoxSize : integer
2244: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2245: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2246: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2247: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2248: Function ReverseBits( Value : Byte) : Byte;
2249: Function ReverseBits1( Value : Shortint) : Shortint;
2250: Function ReverseBits2( Value : Smallint) : Smallint;
2251: Function ReverseBits3( Value : Word) : Word;
2252: Function ReverseBits4( Value : Cardinal) : Cardinal;
2253: Function ReverseBits4( Value : Integer) : Integer;
2254: Function ReverseBits5( Value : Int64) : Int64;
2255: Function ReverseBytes( Value : Word) : Word;
2256: Function ReverseBytes1( Value : Smallint) : Smallint;
2257: Function ReverseBytes2( Value : Integer) : Integer;
2258: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2259: Function ReverseBytes4( Value : Int64) : Int64;
2260: Function ReverseString( const AText : string) : string
2261: Function ReverseDNSLookup(const IPAddr:String;DNSServer:String;Timeout,Retries:Int;var
  HostName:String):Bool;
2262: Function Revert : HRESULT
2263: Function RGB(R,G,B: Byte): TColor;
2264: Function RGB2BGR( const Color : TColor) : TColor

```

```

2265: Function RGB2TColor( R, G, B : Byte ) : TColor
2266: Function RGBToWebColorName( RGB : Integer ) : string
2267: Function RGBToWebColorStr( RGB : Integer ) : string
2268: Function RgbToHtml( Value : TColor ) : string
2269: Function HtmlToRgb(const Value: string): TColor;
2270: Function RightStr( const AStr : String; Len : Integer ) : String
2271: Function RightStr( const AText : AnsiString; const ACount : Integer ) : AnsiString;
2272: Function RightStr( const AText : WideString; const ACount : Integer ) : WideString;
2273: Function ROL( AVal : LongWord; AShift : Byte ) : LongWord
2274: Function ROR( AVal : LongWord; AShift : Byte ) : LongWord
2275: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float ) : TFloatPoint
2276: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2277: Function Round(e : Extended) : Longint;
2278: Function Round64(e: extended): Int64;
2279: Function RoundAt( const Value : string; Position : SmallInt) : string
2280: Function RoundFrequency( const Frequency : Integer ) : Integer
2281: Function RoundInt( Value : Integer; StepSize : Integer ) : Integer
2282: Function RoundPoint( const X, Y : Double ) : TPoint
2283: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double ) : TRect
2284: Function RowCount : Integer
2285: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2286: Function RowRequest( Row : OleVariant; Options : TFetchOptions ) : OleVariant
2287: Function RPos( const ASub, AIn : String; AStart : Integer ) : Integer
2288: Function RRot( const Value : Byte; const Count : TBitRange ) : Byte;
2289: Function RRot1( const Value : Word; const Count : TBitRange ) : Word;
2290: Function RRot2( const Value : Integer; const Count : TBitRange ) : Integer;
2291: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2292: Function RunningProcessesList( const List : TStringList; FullPath : Boolean ) : Boolean
2293: Function S_AddBackSlash( const ADirName : string ) : string
2294: Function S_AllTrim( const cStr : string ) : string
2295: Function S_AtRepl( const cAT, cStr, cRepl : string ) : string
2296: Function S_Cut( const cStr : string; const iLen : integer ) : string
2297: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer ) : integer
2298: Function S_DirExists( const ADir : string ) : Boolean
2299: Function S_Empty( const cStr : string ) : boolean
2300: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer ) : string
2301: Function S_LargeFontsActive : Boolean
2302: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer ) : Extended
2303: Function S_LTrim( const cStr : string ) : string
2304: Function S_ReadNextTextLineFromStream( stream : TStream ) : string
2305: Function S_RepeatChar( const iLen : integer; const AChar : Char ) : String
2306: Function S_Rep1First( const cAT, cStr, cRepl : string ) : string
2307: Function S_RoundDecimal( AValue : Extended; APlaces : Integer ) : Extended
2308: Function S_RTrim( const cStr : string ) : string
2309: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer ) : string
2310: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2311: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd ) : string
2312: Function S_Space( const iLen : integer ) : String
2313: Function S_StrBlanks( const cStr : string; const iLen : integer ) : string
2314: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer ) : string
2315: Function S_StrCRC32( const Text : string ) : LongWORD
2316: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string
2317: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string
2318: Function S_StringToUTF_8( const AString : string ) : string
2319: Function S_StrLBlanks( const cStr : string; const iLen : integer ) : string
2320: function S_StrToReal(const cStr: string; var R: Double): Boolean
2321: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean ) : boolean
2322: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean ) : string
2323: Function S_UTF_8ToString( const AString : string ) : string
2324: Function S_WBox( const AText : string ) : integer
2325: Function SameDate( const A, B : TDateTime ) : Boolean
2326: function SameDate(const A, B: TDateTime): Boolean;
2327: Function SameDateTime( const A, B : TDateTime ) : Boolean
2328: function SameDateTime(const A, B: TDateTime): Boolean;
2329: Function SameFileName( S1, S2 : string ) : Boolean
2330: Function SameText( S1, S2 : string ) : Boolean
2331: Function SameText( const S1: string; const S2: string): Boolean)
2332: Function SameTime( const A, B : TDateTime ) : Boolean
2333: function SameTime(const A, B: TDateTime): Boolean;
2334: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2335: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2336: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2337: Function SampleVariance( const X : TDynFloatArray ) : Float
2338: Function Sar( const Value : Shortint; const Count : TBitRange ) : Shortint;
2339: Function Sar1( const Value : Smallint; const Count : TBitRange ) : Smallint;
2340: Function Sar2( const Value : Integer; const Count : TBitRange ) : Integer;
2341: Function SaveToFile( const AFileName : TFileName ) : Boolean
2342: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2343: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2344: Function ScanF(const aformat: String; const args: array of const): string;
2345: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2346: Function SearchBuf( Buf: PChar; BufLen: Integer; SelStart, SelLength: Integer; SearchString: String; Options: TStringSearchOptions): PChar
2347: Function SearchBuf2( Buf: String; SelStart, SelLength: Integer; SearchString: String; Options: TStringSearchOptions): Integer;
2348: function SearchRecattr: integer;
2349: function SearchRecExcludeAttr: integer;
2350: Function SearchRecFileSize64( const SearchRec : TSearchRec ) : Int64
2351: function SearchRename: string;

```

```

2352: function SearchRecsize: integer;
2353: function SearchRecTime: integer;
2354: function Sec( const X : Extended) : Extended
2355: function Secant( const X : Extended) : Extended
2356: function SecH( const X : Extended) : Extended
2357: function SecondOf( const AValue : TDateTime) : Word
2358: function SecondOfDay( const AValue : TDateTime) : LongWord
2359: function SecondOfTheHour( const AValue : TDateTime) : Word
2360: function SecondOfTheMinute( const AValue : TDateTime) : Word
2361: function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2362: function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2363: function SecondOfTheYear( const AValue : TDateTime) : LongWord
2364: function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2365: function SecondSpan( const ANow, AThen : TDateTime) : Double
2366: function SectionExists( const Section : string) : Boolean
2367: function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2368: function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HRESULT
2369: function Seek(Offset:Longint;Origin:Word):Longint
2370: function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2371: function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2372: function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2373: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2374: function SendBuf( var Buf, Count : Integer) : Integer
2375: function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2376: function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2377: function SendKey( AppName : string; Key : Char) : Boolean
2378: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2379: function SendStream( AStream : TStream) : Boolean
2380: function SendStreamThenDrop( AStream : TStream) : Boolean
2381: function SendText( const S : string) : Integer
2382: function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2383: function SendSerialText(Data: String): cardinal
2384: function Sent : Boolean
2385: function ServicesFilePath: string
2386: function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2387: function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2388: function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2389: function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2390: function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2391: function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2392: function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2393: function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2394: function SetClipboard( NewClipboard : TClipboard) : TClipboard
2395: function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2396: function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2397: function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2398: function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2399: function SetCurrentDir( Dir : string) : Boolean
2400: function SetCurrentDir(const Dir: string): Boolean
2401: function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2402: function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2403: function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2404: function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2405: function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2406: function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2407: function SetEnvironmentVar( const Name, Value : string) : Boolean
2408: function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2409: function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2410: function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2411: function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2412: function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2413: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2414: function SetLocalTime( Value : TDateTime) : boolean
2415: function SetPrecisionTolerance( NewTolerance : Float) : Float
2416: function SetPrinter( NewPrinter : TPrinter) : TPrinter
2417: function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2418: function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2419: function SetSize( libNewSize : Longint) : HRESULT
2420: function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2421: function Sgn( const X : Extended) : Integer
2422: function SHA1(const fileName: string): string;
2423: function SHA256(astr: string; amode: char): string)
2424: function SHA512(astr: string; amode: char): string)
2425: function ShareMemoryManager : Boolean
2426: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2427: function ShellExecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2428: function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2429: function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2430: function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2431: function ShortDateFormat: string;
2432: function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
RTL:Bool;EllipsisWidth:Int):WideString
2433: function ShortTimeFormat: string;
2434: function SHOWMODAL: INTEGER
2435: function ShowWindow(C1: HWND; C2: integer): boolean;
2436: procedure ShowMemory //in Dialog
2437: function ShowMemory2: string;
2438: function ShutDownOS : Boolean

```



```

2439: Function Signe( const X, Y : Extended) : Extended
2440: Function Sign( const X : Extended) : Integer
2441: Function Sin(e : Extended) : Extended;
2442: Function sinc( const x : Double) : Double
2443: Function SinJ( X : Float) : Float
2444: Function Size( const AFileName : String) : Integer
2445: function SizeOf: Longint;
2446: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2447: function SlashSep(const Path, S: String): String
2448: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2449: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2450: Function SmallPoint(X, Y: Integer): TSmallPoint)
2451: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2452: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2453: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2454: Function SoundexProc( const AText, AOther : string) : Boolean
2455: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2456: Function SoundexWord( const AText : string) : Word
2457: Function SourcePos : Longint
2458: function SourcePos:LongInt
2459: Function Split0( Str : string; const substr : string) : TStringList
2460: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2461: Function SQLRequiresParams( const SQL : WideString) : Boolean
2462: Function Sqr(e : Extended) : Extended;
2463: Function Sqrt(e : Extended) : Extended;
2464: Function StartIP : String
2465: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2466: Function StartOfDay( const AYear, AMonth, ADay : Word) : TDateTime;
2467: Function StartOfDay1( const AYear, ADayOfYear : Word) : TDateTime;
2468: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2469: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2470: Function StartOfAYear( const AYear : Word) : TDateTime
2471: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2472: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2473: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2474: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2475: Function StartsStr( const ASubText, AText : string) : Boolean
2476: Function StartsText( const ASubText, AText : string) : Boolean
2477: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2478: Function StartsWith( const str : string; const sub : string) : Boolean
2479: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2480: Function StatusString( StatusCode : Integer) : string
2481: Function StdDev( const Data : array of Double) : Extended
2482: Function Stop : Float
2483: Function StopCount( var Counter : TJclCounter) : Float
2484: Function StoreColumns : Boolean
2485: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2486: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2487: Function StrAlloc( Size : Cardinal) : PChar
2488: function StrAlloc(Size: Cardinal): PChar)
2489: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2490: Function StrBefore1( const sString : string; const sDelimiters: string; out cDelimiter: char): string;
2491: Function StrBufSize( Str : PChar) : Cardinal
2492: function StrBufSize(const Str: PChar): Cardinal)
2493: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2494: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2495: Function StrCat( Dest : PChar; Source : PChar) : PChar
2496: function StrCat(Dest: PChar; const Source: PChar): PChar)
2497: Function StrCharLength( Str : PChar) : Integer
2498: Function StrComp( Str1, Str2 : PChar) : Integer
2499: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2500: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2501: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2502: Function Stream_to_AnsiString( Source : TStream) : ansistring
2503: Function Stream_to_Base64( Source : TStream) : ansistring
2504: Function Stream_to_decimalbytes( Source : TStream) : string
2505: Function Stream2WideString( oStream : TStream) : WideString
2506: Function StreamtoAnsiString( Source : TStream) : ansistring
2507: Function StreamToByte( Source : TStream) : string
2508: Function StreamToDecimalbytes( Source : TStream) : string
2509: Function StreamtoOrd( Source : TStream) : string
2510: Function StreamToString( Source : TStream) : string
2511: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2512: Function StrEmpty( const sString : string) : boolean
2513: Function StrEnd( Str : PChar) : PChar
2514: function StrEnd(const Str: PChar): PChar)
2515: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2516: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2517: Function StrGet(var S : String; I : Integer) : Char;
2518: Function StrGet2(S : String; I : Integer) : Char;
2519: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2520: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2521: Function StrHtmlDecode( const AStr : String) : String
2522: Function StrHtmlEncode( const AStr : String) : String
2523: Function StrToBytes(const Value: String): TBytes;
2524: Function StrIComp( Str1, Str2 : PChar) : Integer
2525: Function StringOfChar(c : char; I : longint) : String;
2526: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2527: Function StringPad(InputStr, FillChar: String; StrLen: Integer; StrJustify: Boolean): String;

```

```

2528: Function StringRefCount(const s: String): integer;
2529: Function StringReplace( S, OldPattern, NewPattern: string; Flags: TReplaceFlags): string
2530: Function JStringReplace( const S, OldPattern, NewPattern: string; Flags: TReplaceFlags): string
2531: Function StringReplace(const SourceString, OldPattern, NewPattern: string; Flags: TReplaceFlags): string;
2532: Function StringRemove( const S, Pattern: string; Flags: TReplaceFlags): string
2533: Function StringToBoolean( const Ps: string): Boolean
2534: function StringToColor(const S: string): TColor
2535: function StringToCursor(const S: string): TCursor;
2536: function StringToGUID(const S: string): TGUID
2537: Function StringTokenizer( const str: string; const delim: string): IStringTokenizer
2538: Function StringToArray( const str: string; const delim: string): TStringDynArray
2539: Function StringWidth( S: string): Integer
2540: Function StrInternetToDateTime( Value: string): TDateTime
2541: Function StrIsDateTime( const Ps: string): Boolean
2542: Function StrIsFloatMoney( const Ps: string): Boolean
2543: Function StrIsInteger( const S: string): Boolean
2544: Function StrLCat( Dest: PChar; Source: PChar; MaxLen: Cardinal): PChar
2545: Function StrLComp( Str1, Str2: PChar; MaxLen: Cardinal): Integer
2546: Function StrLCopy( Dest: PChar; Source: PChar; MaxLen: Cardinal): PChar
2547: Function StrLen( Str: PChar): Cardinal
2548: function StrLen(const Str: PChar): Cardinal
2549: Function StrLessPrefix( const sString: string; const sPrefix: string): string
2550: Function StrLessSuffix( const sString: string; const sSuffix: string): string
2551: Function StrLIComp( Str1, Str2: PChar; MaxLen: Cardinal): Integer
2552: Function StrLower( Str: PChar): PChar
2553: Function StrMove( Dest: PChar; Source: PChar; Count: Cardinal): PChar
2554: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar
2555: Function StrNew( Str: PChar): PChar
2556: function StrNew(const Str: PChar): PChar
2557: Function StrNextChar( Str: PChar): PChar
2558: Function StrPad( const sString: string; const sPad: string; const iLength: integer): string
2559: Function StrParse( var sString: string; const sDelimiters: string): string;
2560: Function StrParse( var sString: string; const sDelimiters: string; out cDelimiter: char): string;
2561: Function StrPas( Str: PChar): string
2562: function StrPas(const Str: PChar): string
2563: Function StrPCopy( Dest: PChar; Source: string): PChar
2564: function StrPCopy(Dest: PChar; const Source: string): PChar
2565: Function StrPLCopy( Dest: PChar; Source: string; MaxLen: Cardinal): PChar
2566: Function StrPos( Str1, Str2: PChar): PChar
2567: Function StrScan(const Str: PChar; Chr: Char): PChar
2568: Function StrRScan(const Str: PChar; Chr: Char): PChar
2569: Function StrToBcd( const AValue: string): TBcd
2570: Function StrToBool( S: string): Boolean
2571: Function StrToBoolDef( S: string; Default: Boolean): Boolean
2572: Function StrToCard( const AStr: String): Cardinal
2573: Function StrToConv( AText: string; out AType: TConvType): Double
2574: Function StrToCurr( S: string): Currency;
2575: function StrToCurr(const S: string): Currency
2576: Function StrToCurrDef( S: string; Default: Currency): Currency;
2577: Function StrToDate( S: string): TDateTime;
2578: function StrToDate(const s: string): TDateTime;
2579: Function StrToDateDef( S: string; Default: TDateTime): TDateTime;
2580: Function StrToDateTime( S: string): TDateTime;
2581: function StrToDateTime(const S: string): TDateTime
2582: Function StrToDateTimeDef( S: string; Default: TDateTime): TDateTime;
2583: Function StrToDay( const ADay: string): Byte
2584: Function StrToFloat( S: string): Extended;
2585: function StrToFloat(s: String): Extended;
2586: Function StrToFloatDef( S: string; Default: Extended): Extended;
2587: function StrToFloatDef(const S: string; const Default: Extended): Extended
2588: Function StrToFloat( S: string): Extended;
2589: Function StrToFloat2( S: string; FormatSettings: TFormatSettings): Extended;
2590: Function StrToFloatDef( S: string; Default: Extended): Extended;
2591: Function StrToFloatDef2(S: string; Default: Extended; FormatSettings: TFormatSettings): Extended;
2592: Function StrToCurr( S: string): Currency;
2593: Function StrToCurr2( S: string; FormatSettings: TFormatSettings): Currency;
2594: Function StrToCurrDef( S: string; Default: Currency): Currency;
2595: Function StrToCurrDef2( S: string; Default: Currency; FormatSettings: TFormatSettings): Currency;
2596: Function StrToTime2( S: string; FormatSettings: TFormatSettings): TDateTime;
2597: Function StrToTimeDef( S: string; Default: TDateTime): TDateTime;
2598: Function StrToTimeDef2( S: string; Default: TDateTime; FormatSettings: TFormatSettings): TDateTime;
2599: Function TryStrToTime( S: string; Value: TDateTime): Boolean;
2600: Function StrToDateTime( S: string): TDateTime;
2601: Function StrToDateTime2( S: string; FormatSettings: TFormatSettings): TDateTime;
2602: Function StrToDateTimeDef( S: string; Default: TDateTime): TDateTime;
2603: Function StrToFloatRegionalIndependent(aValue: String; aDecimalSymbol: Char; aDigitGroupSymbol: Char): Extended
2604: Function StrToInt( S: string): Integer
2605: function StrToInt(s: String): Longint;
2606: Function StrToInt64( S: string): Int64
2607: function StrToInt64(s: String): int64;
2608: Function StrToInt64Def( S: string; Default: Int64): Int64
2609: function StrToInt64Def(const S: string; const Default: Int64): Int64
2610: Function StrToIntDef( S: string; Default: Integer): Integer
2611: function StrToIntDef(const S: string; Default: Integer): Integer
2612: function StrToIntDef(s: String; def: Longint): Longint;
2613: Function StrToMonth( const AMonth: string): Byte
2614: Function StrToTime( S: string): TDateTime;
2615: function StrToTime(const S: string): TDateTime
2616: Function StrToTimeDef( S: string; Default: TDateTime): TDateTime;

```

```

2617: Function StrToWorld( const Value : String ) : Word
2618: Function StrToXmlDate( const DateStr : string; const Format : string ) : string
2619: Function StrToXmlDateTime( const DateStr : string; const Format : string ) : string
2620: Function StrToXmlTime( const TimeStr : string; const Format : string ) : string
2621: Function StrUpper( Str : PChar ) : PChar
2622: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string ) : string
2623: Function Sum( const Data : array of Double ) : Extended
2624: Function SumFloatArray( const B : TDynFloatArray ) : Float
2625: Function SumInt( const Data : array of Integer ) : Integer
2626: Function SumOfSquares( const Data : array of Double ) : Extended
2627: Function SumPairProductFloatArray( const X, Y : TDynFloatArray ) : Float
2628: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float ) : Float
2629: Function SumSquareFloatArray( const B : TDynFloatArray ) : Float
2630: Function Supports( CursorOptions : TCursorOptions ) : Boolean
2631: Function SupportsClipboardFormat( AFormat : Word ) : Boolean
2632: Function SwapWord( w : word ) : word
2633: Function SwapInt( i : integer ) : integer
2634: Function SwapLong( L : longint ) : longint
2635: Function Swap( i : integer ) : integer
2636: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer ) : Extended
2637: Function SyncTime : Boolean
2638: Function SysErrorMessage( ErrorCode : Integer ) : string
2639: function SysErrorMessage( ErrorCode : Integer ) : string
2640: Function SystemTimeToDateTime( SystemTime : TSystemTime ) : TDateTime
2641: function SystemTimeToDateTime( const SystemTime : TSystemTime ) : TDateTime;
2642: Function SysStringLen( const S : WideString ) : Integer; stdcall;
2643: Function TabRect( Index : Integer ) : TRect
2644: Function Tan( const X : Extended ) : Extended
2645: Function TaskMessageDlg( const Title,
    Msg: string; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint ) : Integer;
2646: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx : Longint; DefaultButton : TMsgDlgBtn ) : Integer;
2647: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx : Longint; X, Y : Integer ) : Integer;
2648: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn ) : Integer;
2649: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
    TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string ) : Integer;
2650: Function TaskMessageDlgPosHelp1( const Title, Msg: string; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons;
    HelpCtx: Longint; X, Y: Integer; const HelpFileName: string; DefaultButton: TMsgDlgBtn ) : Integer;
2651: Function TenToY( const Y : Float ) : Float
2652: Function TerminateApp( ProcessID : DWORD; Timeout : Integer ) : TJclTerminateAppResult
2653: Function TerminateTask( Wnd : HWND; Timeout : Integer ) : TJclTerminateAppResult
2654: Function TestBit( const Value : Byte; const Bit : TBitRange ) : Boolean;
2655: Function TestBit2( const Value : Shortint; const Bit : TBitRange ) : Boolean;
2656: Function TestBit3( const Value : Smallint; const Bit : TBitRange ) : Boolean;
2657: Function TestBit4( const Value : Word; const Bit : TBitRange ) : Boolean;
2658: Function TestBit5( const Value : Cardinal; const Bit : TBitRange ) : Boolean;
2659: Function TestBit6( const Value : Integer; const Bit : TBitRange ) : Boolean;
2660: Function TestBit7( const Value : Int64; const Bit : TBitRange ) : Boolean;
2661: Function TestBits( const Value, Mask : Byte ) : Boolean;
2662: Function TestBits1( const Value, Mask : Shortint ) : Boolean;
2663: Function TestBits2( const Value, Mask : Smallint ) : Boolean;
2664: Function TestBits3( const Value, Mask : Word ) : Boolean;
2665: Function TestBits4( const Value, Mask : Cardinal ) : Boolean;
2666: Function TestBits5( const Value, Mask : Integer ) : Boolean;
2667: Function TestBits6( const Value, Mask : Int64 ) : Boolean;
2668: Function TestFDIVInstruction : Boolean
2669: Function TestStreamFormat( Stream : TStream ) : TStreamOriginalFormat
2670: Function TextExtent( const Text : string ) : TSize
2671: function TextHeight( Text : string ) : Integer;
2672: Function TextIsSame( const A1 : string; const A2 : string ) : Boolean
2673: Function TextStartsWith( const S, SubS : string ) : Boolean
2674: function TextToFloat( Buffer: PChar; var Value: Extended; ValueType: TFloatValue ) : Boolean
2675: Function ConvInteger( i : integer ) : string;
2676: Function IntegerToText( i : integer ) : string;
2677: Function TEXTTOSHORTCUT( TEXT : String ) : TSHORTCUT
2678: Function TextWidth( Text : string ) : Integer;
2679: Function ThreadCount : integer
2680: function ThousandSeparator: char;
2681: Function Ticks : Cardinal
2682: Function Time : TDateTime
2683: function Time: TDateTime;
2684: function TimeGetTime: int64;
2685: Function TimeOf( const AValue : TDateTime ) : TDateTime
2686: function TimeSeparator: char;
2687: function TimeStampToDateTime( const TimeStamp : TTimeStamp ) : TDateTime
2688: Function TimeStampToMsecs( TimeStamp : TTimeStamp ) : Comp
2689: function TimeStampToMsecs( const TimeStamp : TTimeStamp ) : Comp
2690: Function TimeToStr( DateTime : TDateTime ) : string;
2691: function TimeToStr( const DateTime : TDateTime ) : string;
2692: Function TimeZoneBias : TDateTime
2693: Function ToCommon( const AValue : Double ) : Double
2694: function ToCommon( const AValue : Double ) : Double;
2695: Function Today : TDateTime
2696: Function ToggleBit( const Value : Byte; const Bit : TBitRange ) : Byte;
2697: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange ) : Shortint;
2698: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange ) : Smallint;
2699: Function ToggleBit3( const Value : Word; const Bit : TBitRange ) : Word;

```

```

2700: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2701: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2702: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2703: function TokenComponentIdent:String
2704: Function TokenFloat : Extended
2705: function TokenFloat:Extended
2706: Function TokenInt : Longint
2707: function TokenInt:LongInt
2708: Function TokenString : string
2709: function TokenString:String
2710: Function TokenSymbolIs( const S : string) : Boolean
2711: function TokenSymbolIs(S:String):Boolean
2712: Function Tomorrow : TDateTime
2713: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2714: Function ToString : string
2715: Function TotalVariance( const Data : array of Double) : Extended
2716: Function Trace2( AURL : string) : string;
2717: Function TrackMenu( Button : TToolButton) : Boolean
2718: Function TRANSLATE( SRC, DEST : PCHAR; TOEM : BOOLEAN) : INTEGER
2719: Function TranslateURI( const URI : string) : string
2720: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2721: Function TransparentStretchBlt( DstDC : HDC; DstX, DstY, DstW, DstH: Integer; SrcDC:HDC; SrcX, SrcY, SrcW,
SrcH: Integer; MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2722: Function Trim( S : string) : string;
2723: Function Trim( S : WideString) : WideString;
2724: Function Trim(s : AnyString) : AnyString;
2725: Function TrimAllOf( ATrim, AText : String) : String
2726: Function TrimLeft( S : string) : string;
2727: Function TrimLeft( S : WideString) : WideString;
2728: function TrimLeft(const S: string): string)
2729: Function TrimRight( S : string) : string;
2730: Function TrimRight( S : WideString) : WideString;
2731: function TrimRight(const S: string): string)
2732: function TrueBoolStrs: array of string
2733: Function Trunc(e : Extended) : Longint;
2734: Function Trunc64(e: extended): Int64;
2735: Function TruncPower( const Base, Exponent : Float) : Float
2736: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2737: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2738: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2739: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2740: Function TryEncodeDateMonthWeek(const AY, AMonth, AWeekOfMonth, ADayOfWeek: Word; var AValue: TDateTime): Boolean
2741: Function TryEncodeDateTime(const AYear, AMonth, ADay, AHour, AMin, ASec, AMilliSecond: Word; out
AValue: TDateTime): Boolean
2742: Function TryEncodeDateWeek(const AY, AWeekOfYear: Word; out AValue: TDateTime; const ADayOfWeek: Word): Boolean
2743: Function TryEncodeDayOfWeekInMonth(const AYear, AMonth, ANthDayOfWeek, ADayOfWeek: Word; out
AVal: TDateTime): Bool
2744: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2745: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2746: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2747: Function TryLock : Boolean
2748: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2749: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
AMilliSecond : Word; out AResult : TDateTime) : Boolean
2750: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2751: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2752: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2753: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2754: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2755: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2756: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2757: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2758: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2759: Function TwoToY( const Y : Float) : Float
2760: Function UCS4StringToWideString( const S : UCS4String) : WideString
2761: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2762: function Unassigned: Variant;
2763: Function UndoLastChange( FollowChange : Boolean) : Boolean
2764: function UniCodeToStr(Value: string): string;
2765: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2766: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2767: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2768: Function UnixPathToDosPath( const Path : string) : string
2769: Function UnixToDateTime( const AValue : Int64) : TDateTime
2770: function UnixToDateTime(U: Int64): TDateTime;
2771: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HRESULT
2772: Function UnlockResource( ResData : HGLOBAL) : LongBool
2773: Function UnlockVolume( var Handle : THandle) : Boolean
2774: Function UnMaskString( Mask, Value : String) : String
2775: function UpCase(ch : Char) : Char;
2776: Function UpCaseFirst( const AStr : string) : string
2777: Function UpCaseFirstWord( const AStr : string) : string
2778: Function UpdateAction( Action : TBasicAction) : Boolean
2779: Function UpdateKind : TUpdateKind
2780: Function UPDATESTATUS : TUPDATESTATUS
2781: Function UpperCase( S : string) : string
2782: Function Uppercase(s : AnyString) : AnyString;
2783: Function URLEnCode( ASrc : string) : string
2784: Function URLEnCode( const ASrc : string) : string

```



```

2785: Function UseRightToLeftAlignment : Boolean
2786: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2787: Function UseRightToLeftReading : Boolean
2788: Function UTF8CharLength( Lead : Char) : Integer
2789: Function UTF8CharSize( Lead : Char) : Integer
2790: Function UTF8Decode( const S : UTF8String) : WideString
2791: Function UTF8Encode( const WS : WideString) : UTF8String
2792: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2793: Function Utf8ToAnsi( const S : UTF8String) : string
2794: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2795: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2796: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2797: Function ValidParentForm(control: TControl): TForm
2798: Function Value : Variant
2799: Function ValueExists( const Section, Ident : string) : Boolean
2800: Function ValueOf( const Key : string) : Integer
2801: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2802: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2803: Function VarArrayFromStrings( Strings : TStrings) : Variant
2804: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2805: Function VarArrayGet( var S : Variant; I : Integer) : Variant;
2806: Function VarFMTBcd : TVarType
2807: Function VarFMTBcdCreatel : Variant;
2808: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2809: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2810: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2811: Function Variance( const Data : array of Double) : Extended
2812: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2813: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2814: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2815: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2816: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2817: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2818: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2819: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2820: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2821: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2822: Function VariantNeg( const V1 : Variant) : Variant
2823: Function VariantNot( const V1 : Variant) : Variant
2824: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2825: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2826: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2827: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2828: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2829: function VarIsEmpty( const V : Variant) : Boolean;
2830: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2831: function VarIsNull( const V : Variant) : Boolean;
2832: Function VarToBcd( const AValue : Variant) : TBcd
2833: function VarType( const V : Variant) : TVarType;
2834: Function VarType( const V : Variant) : TVarType
2835: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2836: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2837: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2838: Function VarIsByRef( const V : Variant) : Boolean
2839: Function VarIsEmpty( const V : Variant) : Boolean
2840: Procedure VarCheckEmpty( const V : Variant)
2841: Function VarIsNull( const V : Variant) : Boolean
2842: Function VarIsClear( const V : Variant) : Boolean
2843: Function VarIsCustom( const V : Variant) : Boolean
2844: Function VarIsOrdinal( const V : Variant) : Boolean
2845: Function VarIsFloat( const V : Variant) : Boolean
2846: Function VarIsNumeric( const V : Variant) : Boolean
2847: Function VarIsStr( const V : Variant) : Boolean
2848: Function VarToStr( const V : Variant) : string
2849: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2850: Function VarToWideStr( const V : Variant) : WideString
2851: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2852: Function VarToDateTime( const V : Variant) : TDateTime
2853: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2854: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2855: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2856: TVariantRelationship, '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2857: Function VarSameValue( const A, B : Variant) : Boolean
2858: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2859: Function VarIsEmptyParam( const V : Variant) : Boolean
2860: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2861: Function VarIsError1( const V : Variant) : Boolean;
2862: Function VarAsError( AResult : HRESULT) : Variant
2863: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2864: Function VarIsArray( const A : Variant) : Boolean;
2865: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2866: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2867: Function VarArrayOf( const Values : array of Variant) : Variant
2868: Function VarArrayRef( const A : Variant) : Variant
2869: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2870: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2871: Function VarArrayDimCount( const A : Variant) : Integer
2872: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2873: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer

```

```

2874: Function VarArrayLock( const A : Variant ) : ___Pointer
2875: Procedure VarArrayUnlock( const A : Variant )
2876: Function VarArrayGet( const A : Variant; const Indices : array of Integer ) : Variant
2877: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer )
2878: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer )
2879: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer )
2880: Function Unassigned : Variant
2881: Function Null : Variant
2882: Function VectorAdd( const V1, V2 : TFloatPoint ) : TFloatPoint
2883: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2884: Function VectorDot( const V1, V2 : TFloatPoint ) : Double
2885: function VectorDot(const V1,V2: TFloatPoint): Double;
2886: Function VectorLengthSqr( const V : TFloatPoint ) : Double
2887: function VectorLengthSqr(const V: TFloatPoint): Double;
2888: Function VectorMult( const V : TFloatPoint; const s : Double ) : TFloatPoint
2889: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2890: Function VectorSubtract( const V1, V2 : TFloatPoint ) : TFloatPoint
2891: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2892: Function Verify( AUserName : String ) : String
2893: Function Versine( X : Float ) : Float
2894: function VersionCheck: boolean;
2895: Function VersionLanguageId( const LangIdRec : TLangIdRec ) : string
2896: Function VersionLanguageName( const LangId : Word ) : string
2897: Function VersionResourceAvailable( const FileName : string ) : Boolean
2898: Function Visible : Boolean
2899: function VolumeID(DriveChar: Char): string
2900: Function WaitFor( const AString : string ) : string
2901: Function WaitFor( const TimeOut : Cardinal ) : TWaitResult
2902: Function WaitFor1 : TWaitResult;
2903: Function WaitForData( Timeout : Longint ) : Boolean
2904: Function WebColorNameToColor( WebColorName : string ) : TColor
2905: Function WebColorStrToColor( WebColor : string ) : TColor
2906: Function WebColorToRGB( WebColor : Integer ) : Integer
2907: Function wGet(aURL, afile: string): boolean;
2908: Function wGet2(aURL, afile: string): boolean; //without file open
2909: Function WebGet(aURL, afile: string): boolean;
2910: Function WebExists: boolean; //alias to isinternet
2911: Function WeekOf( const AValue : TDateTime ) : Word
2912: Function WeekOfTheMonth( const AValue : TDateTime ) : Word;
2913: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word ) : Word;
2914: Function WeekOfTheYear( const AValue : TDateTime ) : Word;
2915: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word ) : Word;
2916: Function WeeksBetween( const ANow, AThen : TDateTime ) : Integer
2917: Function WeeksInAYear( const AYear : Word ) : Word
2918: Function WeeksInYear( const AValue : TDateTime ) : Word
2919: Function WeekSpan( const ANow, AThen : TDateTime ) : Double
2920: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle ) : WideString
2921: Function WideCat( const x, y : WideString ) : WideString
2922: Function WideCompareStr( S1, S2 : WideString ) : Integer
2923: function WideCompareStr(const S1: WideString; const S2: WideString): Integer
2924: Function WideCompareText( S1, S2 : WideString ) : Integer
2925: function WideCompareText(const S1: WideString; const S2: WideString): Integer
2926: Function WideCopy( const src : WideString; index, count : Integer ) : WideString
2927: Function WideDequotedStr( const S : WideString; AQuote : WideChar ) : WideString
2928: Function WideEqual( const x, y : WideString ) : Boolean
2929: function WideFormat(const Format: WideString; const Args: array of const): WideString)
2930: Function WideGreater( const x, y : WideString ) : Boolean
2931: Function WideLength( const src : WideString ) : Integer
2932: Function WideLess( const x, y : WideString ) : Boolean
2933: Function WideLowerCase( S : WideString ) : WideString
2934: function WideLowerCase(const S: WideString): WideString
2935: Function WidePos( const src, sub : WideString ) : Integer
2936: Function WideQuotedStr( const S : WideString; Quote : WideChar ) : WideString
2937: Function WideReplaceStr( const AText, AFromText, ATtoText : WideString ) : WideString
2938: Function WideReplaceText( const AText, AFromText, ATtoText : WideString ) : WideString
2939: Function WideSameStr( S1, S2 : WideString ) : Boolean
2940: function WideSameStr(const S1: WideString; const S2: WideString): Boolean
2941: Function WideSameText( S1, S2 : WideString ) : Boolean
2942: function WideSameText(const S1: WideString; const S2: WideString): Boolean
2943: Function WideStringReplace( const S, OldPattern, NewPattern : WideString; Flags: TReplaceFlags ) : WideString
2944: Function WideStringToUCS4String( const S : WideString ) : UCS4String
2945: Function WideUpperCase( S : WideString ) : WideString
2946: Function Win32BackupFile( const FileName : string; Move : Boolean ) : Boolean
2947: function Win32Check(RetVal: boolean): boolean
2948: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
2949: Function Win32RestoreFile( const FileName : string ) : Boolean
2950: Function Win32Type : TIdWin32Type
2951: Function WinColor( const Color32 : TColor32 ) : TColor
2952: function winexec(FileName: pchar; showCmd: integer): integer;
2953: Function WinExec32( const Cmd : string; const CmdShow : Integer ) : Boolean
2954: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer ) : Cardinal
2955: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer ) : Boolean
2956: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64 ) : Boolean
2957: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64 ) : Boolean
2958: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64 ) : Boolean
2959: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer ) : Boolean
2960: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64 ) : Boolean
2961: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer ) : Boolean
2962: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer ) : Boolean

```

```

2963: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar ) : DWORD
2964: Function WordToStr( const Value : Word ) : String
2965: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint ) : Boolean
2966: Function IntToWorldGridFormatIdent( Value : Longint; var Ident : string ) : Boolean
2967: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
2968: Function WorkArea : Integer
2969: Function WrapText( Line : string; MaxCol : Integer ) : string;
2970: Function WrapText( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer ) : string;
2971: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint ) : HRESULT
2972: function Write(Buffer:String;Count:LongInt):LongInt
2973: Function WriteClient( var Buffer, Count : Integer ) : Integer
2974: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean ) : Cardinal
2975: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string ) : Boolean
2976: Function WriteString( const AString : string ) : Boolean
2977: Function WStrGet(var S : AnyString; I : Integer ) : WideChar;
2978: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list ) : Integer
2979: Function wsprintf( Output : PChar; Format : PChar ) : Integer
2980: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string ) : string
2981: Function XmlTimeToStr( const XmlTime : string; const Format : string ) : string
2982: Function XorDecode( const Key, Source : string ) : string
2983: Function XorEncode( const Key, Source : string ) : string
2984: Function XorString( const Key, Src : ShortString ) : ShortString
2985: Function Yield : Bool
2986: Function YearOf( const AValue : TDateTime ) : Word
2987: Function YearsBetween( const ANow, AThen : TDateTime ) : Integer
2988: Function YearSpan( const ANow, AThen : TDateTime ) : Double
2989: Function Yesterday : TDateTime
2990: Function YesNoDialog(const ACaption, AMsg: string): boolean;
2991: Function ( const Name : string; Proc : TUserFunction)
2992: Function using Special_Scholz from 3.8.5.0
2993: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
2994: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
2995: Function FloatToTime2Dec(value:Extended):Extended;
2996: Function MinToStd(value:Extended):Extended;
2997: Function MinToStdAsString(value:Extended):String;
2998: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
2999: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3000: Function Round2Dec (zahl:Extended):Extended;
3001: Function GetAngle(x,y:Extended):Double;
3002: Function AddAngle(a1,a2:Double):Double;
3003:
3004: *****
3005: unit uPSI_StText;
3006: *****
3007: Function TextSeek( var F : TextFile; Target : LongInt ) : Boolean
3008: Function TextFileSize( var F : TextFile ) : LongInt
3009: Function TextPos( var F : TextFile ) : LongInt
3010: Function TextFlush( var F : TextFile ) : Boolean
3011:
3012: *****
3013: from JvVCLUtils;
3014: *****
3015: { Windows resources (bitmaps and icons) VCL-oriented routines }
3016: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3017: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
  DstY:Integer;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3018: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
  Bitmap: TBitmap; TransparentColor: TColor);
3019: function MakeBitmap(ResID: PChar): TBitmap;
3020: function MakeBitmapID(ResID: Word): TBitmap;
3021: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3022: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3023: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3024: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
  HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3025: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3026: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3027: function AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows, Index: Integer);
3028:
3029: {$IFDEF WIN32}
3030: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
  X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3031:
3032: {$ENDIF}
3033: function MakeIcon(ResID: PChar): TIcon;
3034: function MakeIconID(ResID: Word): TIcon;
3035: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3036: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3037: {$IFDEF WIN32}
3038: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3039: {$ENDIF}
3040: { Service routines }
3041: procedure NotImplemented;
3042: procedure ResourceNotFound(ResID: PChar);
3043: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3044: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3045: function PaletteColor(Color: TColor): Longint;
3046: function WidthOf(R: TRect): Integer;
3047: function HeightOf(R: TRect): Integer;
3048: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3049: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);

```

```

3050: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3051: procedure Delay(MSecs: Longint);
3052: procedure CenterControl(Control: TControl);
3053: Function PaletteEntries( Palette : HPALETTE ) : Integer
3054: Function WindowClassName( Wnd : HWND ) : string
3055: Function ScreenWorkArea : TRect
3056: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3057: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3058: Procedure ActivateWindow( Wnd : HWND)
3059: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3060: Procedure CenterWindow( Wnd : HWND)
3061: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3062: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3063: Function DialogsToPixelsX( Dlgs : Word ) : Word
3064: Function DialogsToPixelsY( Dlgs : Word ) : Word
3065: Function PixelsToDialogsX( Pixs : Word ) : Word
3066: Function PixelsToDialogsY( Pixs : Word ) : Word
3067: {$IFDEF WIN32}
3068: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3069: function MakeVariant(const Values: array of Variant): Variant;
3070: {$ENDIF}
3071: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3072: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3073: function MsgDlg(const Msg: string; AType: TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3074: {$IFDEF CBUILDER}
3075: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3076: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3077: {$ELSE}
3078: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3079: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3080: {$ENDIF CBUILDER}
3081: function IsForegroundTask: Boolean;
3082: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3083: function GetAveCharSize(Canvas: TCanvas): TPoint;
3084: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3085: procedure FreeUnusedOle;
3086: procedure Beep;
3087: function GetWindowsVersionJ: string;
3088: function LoadDLL(const LibName: string): THandle;
3089: function RegisterServer(const ModuleName: string): Boolean;
3090: {$IFDEF WIN32}
3091: function IsLibrary: Boolean;
3092: {$ENDIF}
3093: { Gradient filling routine }
3094: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3095: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
TFillDirection; Colors: Byte);
3096: { String routines }
3097: function GetEnvVar(const VarName: string): string;
3098: function AnsiUpperFirstChar(const S: string): string;
3099: function StringToPChar(var S: string): PChar;
3100: function StrPAlloc(const S: string): PChar;
3101: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
3102: function DropT(const S: string): string;
3103: { Memory routines }
3104: function AllocMemo(Size: Longint): Pointer;
3105: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3106: procedure FreeMemo(var fpBlock: Pointer);
3107: function GetMemoSize(fpBlock: Pointer): Longint;
3108: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3109: {$IFDEF COMPILER5_UP}
3110: procedure FreeAndNil(var Obj);
3111: {$ENDIF}
3112: // from PNGLoader
3113: Function OptimizeForPNG(Image: TLinearBitmap; QuantizationSteps: Integer; TransparentColor: TColor): Integer
3114: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3115: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3116: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3117: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect //TButtons
3118: Function IsAnAllResult( const AModalResult : TModalResult ) : Boolean
3119: Function InitWndProc( HWindow : HWND; Message : Longint; LParam : Longint ) : Longint
3120: AddConstantN('CTL3D_ALL', 'LongWord').SetUInt( $FFFF);
3121: //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3122: //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint ) : Longint
3123: //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3124: Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3125: Procedure SetImeName( Name : TImeName)
3126: Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean ) : Boolean
3127: Function Imm32GetContext( hWnd : HWND ) : HIMC
3128: Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC ) : Boolean
3129: Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword ) : Boolean
3130: Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword ) : Boolean
3131: Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean ) : Boolean
3132: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM ) : Boolean
3133: //Function Imm32SetCompositionFont( hImc : HIMC; lpLogFont : PLOGFONTA ) : Boolean
3134: Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3135: Function Imm32IsIME( hKl : longword ) : Boolean
3136: Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean

```



```

3137: Procedure DragDone( Drop : Boolean)
3138:
3139:
3140: *****added from jvjvclutils
3141: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3142: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3143: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3144: function IsPositiveResult(Value: TModalResult): Boolean;
3145: function IsNegativeResult(Value: TModalResult): Boolean;
3146: function IsAbortResult(const Value: TModalResult): Boolean;
3147: function StripAllFromResult(const Value: TModalResult): TModalResult;
3148: // returns either BrightColor or DarkColor depending on the luminance of AColor
3149: // This function gives the same result (AFAIK) as the function used in Windows to
3150: // calculate the desktop icon text color based on the desktop background color
3151: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3152: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3153:
3154: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3155:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3156:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3157:   var LinkName: string; Scale: Integer = 100); overload;
3158: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3159:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3160:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3161:   var LinkName: string; Scale: Integer = 100); overload;
3162: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3163:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3164: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3165:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3166:   Scale: Integer = 100): string;
3167: function HTMLPlainText(const Text: string): string;
3168: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3169:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3170: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3171:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3172: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3173: function HTMLPrepareText(const Text: string): string;
3174:
3175: ***** uPSI_JvAppUtils;
3176: Function GetDefaultSection( Component : TComponent) : string
3177: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3178: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3179: Function GetDefaultIniName : string
3180: ///'OnGetDefaultIniName','TOnGetDefaultIniName');
3181: Function GetDefaultIniRegKey : string
3182: Function FindForm( FormClass : TFormClass) : TForm
3183: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3184: Function ShowDialog( FormClass : TFormClass) : Boolean
3185: ///Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3186: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3187: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3188: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3189: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3190: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3191: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3192: Function StrToIniStr( const Str : string) : string
3193: Function IniStrToStr( const Str : string) : string
3194: Function IniReadString( IniFile : TObjet; const Section, Ident, Default : string) : string
3195: Procedure IniWriteString( IniFile : TObjet; const Section, Ident, Value : string)
3196: Function IniReadInteger( IniFile : TObjet; const Section, Ident : string; Default : Longint) : Longint
3197: Procedure IniWriteInteger( IniFile : TObjet; const Section, Ident : string; Value : Longint)
3198: Function IniReadBool( IniFile : TObjet; const Section, Ident : string; Default : Boolean) : Boolean
3199: Procedure IniWriteBool( IniFile : TObjet; const Section, Ident : string; Value : Boolean)
3200: Procedure IniReadSections( IniFile : TObjet; Strings : TStrings)
3201: Procedure IniEraseSection( IniFile : TObjet; const Section : string)
3202: Procedure IniDeleteKey( IniFile : TObjet; const Section, Ident : string)
3203: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3204: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3205: Procedure AppTaskbarIcons( AppOnly : Boolean)
3206: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObjet; const Section : string)
3207: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObjet; const Section : string)
3208: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObjet)
3209: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObjet)
3210: ***** uPSI_JvDBUtils;
3211: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
3212: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3213: Procedure RefreshQuery( Query : TDataSet)
3214: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3215: Function DataSetSectionName( DataSet : TDataSet) : string
3216: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObjet; const Section : string)
3217: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObjet;const Section:string;RestoreVisible:Bool)
3218: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
Options: TLocateOptions) : Boolean
3219: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3220: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3221: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3222: Function ConfirmDelete : Boolean
3223: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3224: Procedure CheckRequiredField( Field : TField)

```

```

3225: Procedure CheckRequiredFields( const Fields : array of TField)
3226: Function DateToSQL( Value : TDateTime) : string
3227: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3228: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3229: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
HighEmpty:Double;Inclusive:Bool):string
3230: Function StrMaskSQL( const Value : string) : string
3231: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3232: Function FormatAnsiSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3233: Procedure _DBError( const Msg : string)
3234: Const('TrueExpr','String '0=0
3235: Const('sdfStandard16','String ''''mm''/'''dd''/'''yyyy''''''
3236: Const('sdfStandard32','String ''''dd/mm/yyyy''''''
3237: Const('sdfOracle','String 'TO_DATE('''dd/mm/yyyy''', 'DD/MM/YYYY')'
3238: Const('sdfInterbase','String 'CAST('''mm''/'''dd''/'''yyyy'''' AS DATE)'
3239: Const('sdfMSSQL','String 'CONVERT(datetime, ''mm''/'''dd''/'''yyyy''', 103)'
3240: AddTypeS('Largeint', 'Longint
3241: addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3242: 'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3243: 'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3244: 'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutOfRecordRange, '+
3245: 'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportedError);
3246: (*-----*)
3247: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3248: begin
3249: Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3250: Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3251: Function JIniReadString( const FileName, Section, Line : string) : string
3252: Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3253: Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3254: Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3255: Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3256: Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3257: end;
3258:
3259: (* === compile-time registration functions === *)
3260: (*-----*)
3261: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3262: begin
3263: 'UnixTimeStart','LongInt'( 25569);
3264: Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3265: Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3266: Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3267: Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3268: Function CenturyOfDate( const DateTime : TDateTime) : Integer
3269: Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3270: Function DayOfDate( const DateTime : TDateTime) : Integer
3271: Function MonthOfDate( const DateTime : TDateTime) : Integer
3272: Function YearOfDate( const DateTime : TDateTime) : Integer
3273: Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3274: Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3275: Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3276: Function HourOfTime( const DateTime : TDateTime) : Integer
3277: Function MinuteOfTime( const DateTime : TDateTime) : Integer
3278: Function SecondOfTime( const DateTime : TDateTime) : Integer
3279: Function GetISOYearNumberOfDays( const Year : Word) : Word
3280: Function IsISOLongYear( const Year : Word) : Boolean;
3281: Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3282: Function ISODayOfWeek( const DateTime : TDateTime) : Word
3283: Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3284: Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3285: Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3286: Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3287: Function JIsLeapYear( const Year : Integer) : Boolean;
3288: Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3289: Function JDaysInMonth( const DateTime : TDateTime) : Integer
3290: Function Make4DigitYear( Year, Pivot : Integer) : Integer
3291: Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3292: Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3293: Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3294: Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3295: Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3296: Function HoursToMsecs( Hours : Integer) : Integer
3297: Function MinutesToMsecs( Minutes : Integer) : Integer
3298: Function SecondsToMsecs( Seconds : Integer) : Integer
3299: Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3300: Function TimeOfDateTimeToMsecs( DateTime : TDateTime) : Integer
3301: Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3302: Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3303: Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3304: Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3305: Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3306: Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3307: Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3308: Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3309: Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3310: Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3311: Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3312: Function DosDateTimeToStr( DateTime : Integer) : string

```

```

3313: Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3314: Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3315: Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3316: Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3317: Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3318: Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3319: Function FileTimeToStr( const FileTime : TFileTime) : string
3320: Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3321: Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3322: Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3323: Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3324: Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3325: Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3326: Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3327: TJclUnixTime32, 'Longword
3328: Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3329: Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3330: Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3331: Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3332: Function JNullStamp : TTimeStamp
3333: Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3334: Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3335: Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3336: Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3337: Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3338: Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3339: Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3340: Function LastWeekDay1( const Year, Month : Integer) : Integer;
3341: Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3342: Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3343: Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3344: Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3345: Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3346: Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3347: Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3348: Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3349: Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3350: FindClass('TOBJECT'),'EJclDateTimeError
3351: end;
3352:
3353: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3354: begin
3355:   Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3356:   Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3357:   Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3358:   Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3359:   Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3360:   TJclKillLevel, '( klNormal, klNoSignal, klTimeout )
3361:   Function ExitWindows( ExitCode : Cardinal) : Boolean
3362:   Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3363:   Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3364:   Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3365:   Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3366:   Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3367:   Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3368:   Function ShutDownDialog( const DialogMessage:string;Timeout:DWORD;Force,Reboot:Boolean):Boolean;;
3369:   Function ShutDownDialog1(const MachineName,DialogMessage:string;Timeout:DWORD;Force,Reboot:Bool):Bool;
3370:   Function AbortShutDown : Boolean;
3371:   Function AbortShutDown1( const MachineName : string) : Boolean;
3372:   TJclAllowedPowerOperation, '( apoHibernate, apoShutDown, apoSuspend )
3373:   TJclAllowedPowerOperations, 'set of TJclAllowedPowerOperation
3374:   Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3375:   FindClass('TOBJECT'),'EJclCreateProcessError
3376:   Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3377:   Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
Environment:PChar);
3378:   // with Add(EJclCreateProcessError) do
3379:   end;
3380:
3381:
3382: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3383: begin
3384:   //'AnsiSigns','Set').SetSet(['-', '+']);
3385:   'C1_UPPER','LongWord( $0001);
3386:   'C1_LOWER','LongWord( $0002);
3387:   'C1_DIGIT','LongWord').SetUInt( $0004);
3388:   'C1_SPACE','LongWord').SetUInt( $0008);
3389:   'C1_PUNCT','LongWord').SetUInt( $0010);
3390:   'C1_CNTRL','LongWord').SetUInt( $0020);
3391:   'C1_BLANK','LongWord').SetUInt( $0040);
3392:   'C1_XDIGIT','LongWord').SetUInt( $0080);
3393:   'C1_ALPHA','LongWord').SetUInt( $0100);
3394:   AnsiChar, 'Char
3395:   Function StrIsAlpha( const S : AnsiString) : Boolean
3396:   Function StrIsAlphaNum( const S : AnsiString) : Boolean
3397:   Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3398:   Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3399:   Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3400:   Function StrIsDigit( const S : AnsiString) : Boolean

```

```

3401: Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet ) : Boolean
3402: Function StrSame( const S1, S2 : AnsiString ) : Boolean
3403: //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar ) : AnsiString
3404: Function StrCharPosLower( const S : AnsiString; CharPos : Integer ) : AnsiString
3405: Function StrCharPosUpper( const S : AnsiString; CharPos : Integer ) : AnsiString
3406: Function StrDoubleQuote( const S : AnsiString ) : AnsiString
3407: Function StrEnsureNoPrefix( const Prefix, Text : AnsiString ) : AnsiString
3408: Function StrEnsureNoSuffix( const Suffix, Text : AnsiString ) : AnsiString
3409: Function StrEnsurePrefix( const Prefix, Text : AnsiString ) : AnsiString
3410: Function StrEnsureSuffix( const Suffix, Text : AnsiString ) : AnsiString
3411: Function StrEscapedToString( const S : AnsiString ) : AnsiString
3412: Function JStrLower( const S : AnsiString ) : AnsiString
3413: Procedure StrLowerInPlace( var S : AnsiString )
3414: ///Procedure StrLowerBuff( S : PAnsiChar)
3415: Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3416: Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar ) : AnsiString
3417: Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar ) : AnsiString
3418: Function StrProper( const S : AnsiString ) : AnsiString
3419: //Procedure StrProperBuff( S : PAnsiChar)
3420: Function StrQuote( const S : AnsiString; C : AnsiChar ) : AnsiString
3421: Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3422: Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3423: Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3424: Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar ) : AnsiString
3425: Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3426: Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3427: Function StrRepeat( const S : AnsiString; Count : Integer ) : AnsiString
3428: Function StrRepeatLength( const S : AnsiString; const L : Integer ) : AnsiString
3429: Function StrReverse( const S : AnsiString ) : AnsiString
3430: Procedure StrReverseInPlace( var S : AnsiString )
3431: Function StrSingleQuote( const S : AnsiString ) : AnsiString
3432: Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet ) : AnsiString
3433: Function StrStringToEscaped( const S : AnsiString ) : AnsiString
3434: Function StrStripNonNumberChars( const S : AnsiString ) : AnsiString
3435: Function StrToHex( const Source : AnsiString ) : AnsiString
3436: Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar ) : AnsiString
3437: Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3438: Function StrTrimCharRight( const S : AnsiString; C : AnsiChar ) : AnsiString
3439: Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3440: Function StrTrimQuotes( const S : AnsiString ) : AnsiString
3441: Function JStrUpper( const S : AnsiString ) : AnsiString
3442: Procedure StrUpperInPlace( var S : AnsiString )
3443: ///Procedure StrUpperBuff( S : PAnsiChar)
3444: Function StrOemToAnsi( const S : AnsiString ) : AnsiString
3445: Function StrAnsiToOem( const S : AnsiString ) : AnsiString
3446: Procedure StrAddRef( var S : AnsiString )
3447: Function StrAllocSize( const S : AnsiString ) : Longint
3448: Procedure StrDecRef( var S : AnsiString )
3449: //Function StrLen( S : PAnsiChar ) : Integer
3450: Function StrLength( const S : AnsiString ) : Longint
3451: Function StrRefCount( const S : AnsiString ) : Longint
3452: Procedure StrResetLength( var S : AnsiString )
3453: Function StrCharCount( const S : AnsiString; C : AnsiChar ) : Integer
3454: Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet ) : Integer
3455: Function StrStrCount( const S, SubS : AnsiString ) : Integer
3456: Function StrCompare( const S1, S2 : AnsiString ) : Integer
3457: Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer ) : Integer
3458: //Function StrFillChar( const C : AnsiChar; Count : Integer ) : AnsiString;
3459: Function StrFillChar( const C : Char; Count : Integer ) : AnsiString;
3460: Function StrFillChar(const C: Char; Count: Integer): string'';
3461: Function IntFillChar(const I: Integer; Count: Integer): string'';
3462: Function ByteFillChar(const B: Byte; Count: Integer): string'';
3463: Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;'';
3464: Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3465: Function StrFind( const Substr, S : AnsiString; const Index : Integer ) : Integer
3466: //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString ) : Boolean
3467: Function StrIndex( const S : AnsiString; const List : array of AnsiString ) : Integer
3468: Function StrILastPos( const SubStr, S : AnsiString ) : Integer
3469: Function StrIPos( const SubStr, S : AnsiString ) : Integer
3470: Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString ) : Boolean
3471: Function StrLastPos( const SubStr, S : AnsiString ) : Integer
3472: Function StrMatch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3473: Function StrMatches( const Substr, S : AnsiString; const Index : Integer ) : Boolean
3474: Function StrNIPos( const S, SubStr : AnsiString; N : Integer ) : Integer
3475: Function StrNPos( const S, SubStr : AnsiString; N : Integer ) : Integer
3476: Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString ) : Integer
3477: Function StrSearch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3478: //Function StrAfter( const SubStr, S : AnsiString ) : AnsiString
3479: //Function StrBefore( const SubStr, S : AnsiString ) : AnsiString
3480: Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar ) : AnsiString
3481: Function StrChopRight( const S : AnsiString; N : Integer ) : AnsiString
3482: Function StrLeft( const S : AnsiString; Count : Integer ) : AnsiString
3483: Function StrMid( const S : AnsiString; Start, Count : Integer ) : AnsiString
3484: Function StrRestOf( const S : AnsiString; N : Integer ) : AnsiString
3485: Function StrRight( const S : AnsiString; Count : Integer ) : AnsiString
3486: Function CharEqualNoCase( const C1, C2 : AnsiChar ) : Boolean
3487: Function CharIsAlpha( const C : AnsiChar ) : Boolean
3488: Function CharIsAlphaNum( const C : AnsiChar ) : Boolean
3489: Function CharIsBlank( const C : AnsiChar ) : Boolean

```



```

3490: Function CharIsControl( const C : AnsiChar) : Boolean
3491: Function CharIsDelete( const C : AnsiChar) : Boolean
3492: Function CharIsDigit( const C : AnsiChar) : Boolean
3493: Function CharIsLower( const C : AnsiChar) : Boolean
3494: Function CharIsNumberChar( const C : AnsiChar) : Boolean
3495: Function CharIsPrintable( const C : AnsiChar) : Boolean
3496: Function CharIsPunctuation( const C : AnsiChar) : Boolean
3497: Function CharIsReturn( const C : AnsiChar) : Boolean
3498: Function CharIsSpace( const C : AnsiChar) : Boolean
3499: Function CharIsUpper( const C : AnsiChar) : Boolean
3500: Function CharIsWhiteSpace( const C : AnsiChar) : Boolean
3501: Function CharType( const C : AnsiChar) : Word
3502: Function CharHex( const C : AnsiChar) : Byte
3503: Function CharLower( const C : AnsiChar) : AnsiChar
3504: Function CharUpper( const C : AnsiChar) : AnsiChar
3505: Function CharToggleCase( const C : AnsiChar) : AnsiChar
3506: Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3507: Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3508: Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer) : Integer
3509: Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar) : Integer
3510: Procedure StrToTStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3511: Procedure StrToTStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3512: Function StringsToStr(const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool):AnsiString;
3513: Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean)
3514: Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean)
3515: Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean)
3516: Function AddStringToTStrings(const S:AnsiString;Strings:TStrings; const Unique:Boolean):Boolean
3517: Function BooleanToStr( B : Boolean) : AnsiString
3518: Function FileToString( const FileName : AnsiString) : AnsiString
3519: Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean)
3520: Function StrToken( var S : AnsiString; Separator : AnsiChar) : AnsiString
3521: Procedure StrTokens( const S : AnsiString; const List : TStrings)
3522: Procedure StrTokenToTStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings)
3523: Function StrWord( var S : PAnsiChar; out Word : AnsiString) : Boolean
3524: Function StrToFloatSafe( const S : AnsiString) : Float
3525: Function StrToIntSafe( const S : AnsiString) : Integer
3526: Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer);
3527: Function ArrayOf( List : TStrings) : TDynStringArray;
3528: EJclStringError', 'EJclError
3529: function IsClass(Address: TObject): Boolean;
3530: function IsObject(Address: TObject): Boolean;
3531: // Console Utilities
3532: //function ReadKey: Char;
3533: function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3534: function JclGUIDToString(const GUID: TGUID): string;
3535: function JclStringToGUID(const S: string): TGUID;
3536:
3537: end;
3538:
3539:
3540: ***** uPSI_JvDBUtil;
3541: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const
Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
3542: Function GetQueryResult( const DatabaseName, SQL : string) : Variant
3543: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant;
const AResultName : string) : Variant
3544: //Function StrFieldDesc( Field : FLDDesc) : string
3545: Function Var2Type( V : Variant; const VarType : Integer) : Variant
3546: Procedure CopyRecord( DataSet : TDataSet)
3547: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string;
MasterField : Word; ModOp, DelOp : RINTQual)
3548: Procedure AddMasterPassword( Table : TTable; pswd : string)
3549: Procedure PackTable( Table : TTable)
3550: Procedure PackEncryptedTable( Table : TTable; pswd : string)
3551: Function EncodeQuotes( const S : string) : string
3552: Function Cmp( const S1, S2 : string) : Boolean
3553: Function SubStr( const S : string; const Index : Integer; const Separator : string) : string
3554: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string) : string
3555: Function ReplaceString( S : string; const OldPattern, NewPattern : string) : string
3556: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer)
3557: *****uPSI_JvJvBDEUtils;*****
3558: //JvBDEUtils
3559: Function CreateDbLocate : TJvLocateObject
3560: //Function CheckOpen( Status : DBIResult) : Boolean
3561: Procedure FetchAllRecords( DataSet : TBDEDataSet)
3562: Function TransActive( Database : TDatabase) : Boolean
3563: Function AsyncQrySupported( Database : TDatabase) : Boolean
3564: Function GetQuoteChar( Database : TDatabase) : string
3565: Procedure ExecuteQuery( const DbName, QueryText : string)
3566: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string)
3567: Function FieldLogicMap( FldType : TFieldType) : Integer
3568: Function FieldSubtypeMap( FldType : TFieldType) : Integer Value : string; Buffer : Pointer)
3569: Function GetAliasPath( const AliasName : string) : string
3570: Function IsDirectory( const DatabaseName : string) : Boolean
3571: Function GetBdeDirectory : string
3572: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) : Boolean
3573: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3574: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3575: Function DataSetRecNo( DataSet : TDataSet) : Longint

```

```

3576: Function DataSetRecordCount( DataSet : TDataSet ) : Longint
3577: Function DataSetPositionStr( DataSet : TDataSet ) : string
3578: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean)
3579: Function CurrentRecordDeleted( DataSet : TBDEDataSet ) : Boolean
3580: Function IsFilterApplicable( DataSet : TDataSet ) : Boolean
3581: Function IsBookmarkStable( DataSet : TBDEDataSet ) : Boolean
3582: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)
3583: Procedure RestoreIndex( Table : TTable)
3584: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)
3585: Procedure PackTable( Table : TTable)
3586: Procedure ReindexTable( Table : TTable)
3587: Procedure BdeFlushBuffers
3588: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer ) : Pointer
3589: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3590: Procedure DbNotSupported
3591: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
  AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3592: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
  AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter:Char;MaxRecordCount:Longint);
3593: Procedure
  ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3594: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3595: *****uPSI_JvDateUtil;
3596: function CurrentYear: Word;
3597: function IsLeapYear(AYear: Integer): Boolean;
3598: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3599: function FirstDayOfPrevMonth: TDateTime;
3600: function LastDayOfPrevMonth: TDateTime;
3601: function FirstDayOfNextMonth: TDateTime;
3602: function ExtractDay(ADate: TDateTime): Word;
3603: function ExtractMonth(ADate: TDateTime): Word;
3604: function ExtractYear(ADate: TDateTime): Word;
3605: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3606: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3607: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3608: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3609: function ValidDate(ADate: TDateTime): Boolean;
3610: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3611: function MonthsBetween(Date1, Date2: TDateTime): Double;
3612: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3613: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3614: function DaysBetween(Date1, Date2: TDateTime): Longint;
3615: { The same as previous but if Date2 < Date1 result = 0 }
3616: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3617: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3618: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3619: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3620: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3621: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3622: { String to date conversions }
3623: function GetDateOrder(const DateFormat: string): TDateOrder;
3624: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3625: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3626: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3627: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3628: function DefDateFormat(FourDigitYear: Boolean): string;
3629: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3630: -----
3631: ***** JvUtils;*****
3632: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3633: function GetWordOnPos(const S: string; const P: Integer): string;
3634: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3635: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3636: { SubStr returns substring from string, S, separated with Separator string}
3637: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3638: { SubStrEnd same to previous function but Index numerated from the end of string }
3639: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3640: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3641: function SubWord(P: PChar; var P2: PChar): string;
3642: { NumberByWord returns the text representation of
  the number, N, in normal russian language. Was typed from Monitor magazine }
3644: function NumberByWord(const N: Longint): string;
3645: // function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3646: //the symbol Pos is pointed. Lines separated with #13 symbol }
3647: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3648: { GetXYByPos is same to previous function, but returns X position in line too}
3649: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3650: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3651: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3652: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3653: function ConcatSep(const S, S2, Separator: string): string;
3654: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3655: function ConcatLeftSep(const S, S2, Separator: string): string;
3656: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3657: function MinimizeString(const S: string; const MaxLen: Integer): string;
3658: { Next 4 function for russian chars transliterating.
  This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3660: procedure Dos2Win(var S: string);
3661: procedure Win2Dos(var S: string);

```

```

3662: function Dos2WinRes(const S: string): string;
3663: function Win2DosRes(const S: string): string;
3664: function Win2Koi(const S: string): string;
3665: { Spaces returns string consists on N space chars }
3666: function Spaces(const N: Integer): string;
3667: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3668: function AddSpaces(const S: string; const N: Integer): string;
3669: { function LastDate for russian users only } { returns date relative to current date: ' ' }
3670: function LastDate(const Dat: TDateTime): string;
3671: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3672: function CurrencyToStr(const Cur: currency): string;
3673: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3674: function Cmp(const S1, S2: string): Boolean;
3675: { StringCat add S2 string to S1 and returns this string }
3676: function StringCat(var S1: string; S2: string): string;
3677: { HasChar returns True, if Char, Ch, contains in string, S }
3678: function HasChar(const Ch: Char; const S: string): Boolean;
3679: function HasAnyChar(const Chars: string; const S: string): Boolean;
3680: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3681: function CountOfChar(const Ch: Char; const S: string): Integer;
3682: function DefStr(const S: string; Default: string): string;
3683: {**** files routines}
3684: { GetWinDir returns Windows folder name }
3685: function GetWinDir: TFileName;
3686: function GetSysDir: string;
3687: { GetTempDir returns Windows temporary folder name }
3688: function GetTempDir: string;
3689: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3690: function GenTempFileName(FileName: string): string;
3691: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3692: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3693: { ClearDir clears folder Dir }
3694: function ClearDir(const Dir: string): Boolean;
3695: { DeleteDir clears and then delete folder Dir }
3696: function DeleteDir(const Dir: string): Boolean;
3697: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3698: function FileEquMask(FileName, Mask: TFileName): Boolean;
3699: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3700:   Masks must be separated with comma (',';) }
3701: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3702: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3703: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3704: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3705: { FileGetInfo fills SearchRec record for specified file attributes}
3706: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3707: { HasSubFolder returns True, if folder APath contains other folders }
3708: function HasSubFolder(APath: TFileName): Boolean;
3709: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3710: function IsEmptyFolder(APath: TFileName): Boolean;
3711: { AddSlash add slash Char to Dir parameter, if needed }
3712: procedure AddSlash(var Dir: TFileName);
3713: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3714: function AddSlash2(const Dir: TFileName): string;
3715: { AddPath returns FileName with Path, if FileName not contain any path }
3716: function AddPath(const FileName, Path: TFileName): TFileName;
3717: function AddPaths(const PathList, Path: string): string;
3718: function ParentPath(const Path: TFileName): TFileName;
3719: function FindInPath(const FileName, PathList: string): TFileName;
3720: function FindInPaths(const fileName,paths: String): String;
3721: {$IFDEF BCB1}
3722: { BrowseForFolder displays Browse For Folder dialog }
3723: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3724: {$ENDIF BCB1}
3725: function BrowseForFolder(const ATitle: string; AllowCreate: Boolean; var ADirectory: string;
  AHelpContext: THelpContext): Boolean
3726: function BrowseForComputer(const ATitle: string; AllowCreate: Boolean; var ADirectory: string;
  AHelpContext: THelpContext): Boolean
3727: function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3728: function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3729:
3730: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3731: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3732: { HasParam returns True, if program running with specified parameter, Param }
3733: function HasParam(const Param: string): Boolean;
3734: function HasSwitch(const Param: string): Boolean;
3735: function Switch(const Param: string): string;
3736: { ExePath returns ExtractFilePath(ParamStr(0)) }
3737: function ExePath: TFileName;
3738: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3739: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3740: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3741: {**** Graphic routines }
3742: { TTFontSelected returns True, if True Type font is selected in specified device context }
3743: function TTFontSelected(const DC: HDC): Boolean;
3744: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3745: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3746: {**** Windows routines }
3747: { SetWindowTop put window to top without recreating window }
3748: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);

```

```

3749: {**** other routines }
3750: { KeyPressed returns True, if Key VK is now pressed }
3751: function KeyPressed(VK: Integer): Boolean;
3752: procedure SwapInt(var Int1, Int2: Integer);
3753: function IntPower(Base, Exponent: Integer): Integer;
3754: function ChangeTopException(E: TObject): TObject;
3755: function StrToBool(const S: string): Boolean;
3756: {$IFDEF COMPILER3_UP}
3757: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3758:   Length of MaxLen bytes. The compare operation is controlled by the
3759:   current Windows locale. The return value is the same as for CompareStr. }
3760: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3761: function AnsiStrIComp(S1, S2: PChar): Integer;
3762: {$ENDIF}
3763: function Var2Type(V: Variant; const VarType: Integer): Variant;
3764: function VarToInt(V: Variant): Integer;
3765: function VarToFloat(V: Variant): Double;
3766: { following functions are not documented because they are don't work properly , so don't use them }
3767: function ReplaceSokrl(S: string; const Word, Frase: string): string;
3768: { ReplaceSokrl is full equal to ReplaceString function - only for compatibility - don't use }
3769: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3770: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3771: function GetParameter: string;
3772: function GetLongFileName(FileName: string): string;
3773: {* from FileCtrl}
3774: function DirectoryExists(const Name: string): Boolean;
3775: procedure ForceDirectories(Dir: string);
3776: {# from FileCtrl}
3777: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3778: function GetComputerID: string;
3779: function GetComputerName: string;
3780: {**** string routines }
3781: { ReplaceAllSokr searches for all substrings, Words, in a string, S, and replaces them with Frases with the
   same Index. Also see RAUtilsW.ReplaceSokrl function }
3782: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3783: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3784:   in the list, Words, and if founds, replaces this Word with string from another list, Frases, with the
   same Index, and then update NewSelStart variable }
3785: function ReplaceSokr(S: string; PosBeg, Len: Integer; Words, Frases: TStrings; var NewSelStart: Integer): string;
3786: { CountOfLines calculates the lines count in a string, each line separated from another with CrLf sequence }
3787: function CountOfLines(const S: string): Integer;
3788: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3789: procedure DeleteEmptyLines(Ss: TStrings);
3790: { SQLAddWhere adds or modifies existing where-statement, where, to the strings, SQL.
3791:   Note: If strings SQL already contains where-statement, it must be started on begining of any line }
3792: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3793: {**** files routines - }
3794: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3795:   Resource can be compressed using MS Compress program}
3796: function ResSaveToFile(const Typ, Name: string; const Compressed: Boolean; const FileName: string): Boolean;
3797: function ResSaveToFileEx(Inst: HINST; Typ, Name: PChar; const Compressed: Boolean; const FileName: string): Boolean;
3798: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3799: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3800: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3801: { IniReadSection read section, Section, from ini-file,
3802:   IniFileName, into strings, Ss. This function reads ALL strings from specified section.
3803:   Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
3804: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3805: { LoadTextFile load text file, FileName, into string }
3806: function LoadTextFile(const FileName: TFileName): string;
3807: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3808: { ReadFolder reads files list from disk folder, Folder, that are equal Mask, into strings, FileList}
3809: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3810: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3811: {$IFDEF COMPILER3_UP}
3812: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3813: function TargetFileName(const FileName: TFileName): TFileName;
3814: { return filename ShortCut linked to }
3815: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3816: {$ENDIF COMPILER3_UP}
3817: {**** Graphic routines - }
3818: { LoadIcoToImage loads two icons from resource named NameRes, into two image lists ALarge and ASmall}
3819: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3820: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3821: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3822: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3823: function RATextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: string; const CalcHeight: Boolean): Integer;
3824: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3825: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3826: { Cinema draws some visual effect }
3827: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3828: { Roughed fills rect with special 3D pattern }
3829: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3830: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
   and height, AWidth, AHeight and placed on a specified Index, Index in the source bitmap }
3831: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3832: { TextWidth calculate text with for writing using standard desktop font }
3833: function TextWidth(AStr: string): Integer;
3834: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }

```



```

3835: function DefineCursor(Identifier: PChar): TCursor;
3836: {**** other routines - }
3837: { FindFormByClass returns first form specified class, FormClass, owned by Application global variable }
3838: function FindFormByClass(FormClass: TFormClass): TForm;
3839: function FindFormByClassName(FormClassName: string): TForm;
3840: { FindByTag returns the control with specified class, ComponentClass, from WinControl.Controls property,
3841:   having Tag property value, equaled to Tag parameter }
3842: function FindByTag(WinControl: TWinControl; ComponentClass: TComponentClass; const Tag: Integer): TComponent;
3843: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3844: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3845: { RBTAG searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3846: function RBTAG(Parent: TWinControl): Integer;
3847: { AppMinimized returns True, if Application is minimized }
3848: function AppMinimized: Boolean;
3849: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3850:   if Caption parameter = '', it replaced with Application.Title }
3851: function MessageBoxJ(const Msg: string; Caption: string; const Flags: Integer): Integer;
3852: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3853:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3854: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3855:   Buttons: TMsgDlgButtons; DefButton: TMsgDlgBtn; HelpContext: Integer; Control: TWinControl): Integer;
3856: { Delay stop program execution to MSec msec }
3857: procedure Delay(MSec: Longword);
3858: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3859: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3860: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3861: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3862: function PanelBorder(Panel: TCustomPanel): Integer;
3863: function Pixels(Control: TControl; APixels: Integer): Integer;
3864: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3865: procedure Error(const Msg: string);
3866: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3867:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3868: {ex. Text parameter: 'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>' }
3869: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3870:   const HideSelColor: Boolean): string;
3871: function ItemHtWidth(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3872:   const HideSelColor: Boolean): Integer;
3873: function ItemHtPlain(const Text: string): string;
3874: { ClearList - clears list of TObject }
3875: procedure ClearList(List: TList);
3876: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
3877: procedure ClipboardToMemStream(MemStream: TMemoryStream; const Format: Word);
3878: { RTTI support }
3879: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3880: function GetPropStr(Obj: TObject; const PropName: string): string;
3881: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3882: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3883: procedure PrepareIniSection(SS: TStrings);
3884: { following functions are not documented because they are don't work properly, so don't use them }
3885: {$IFDEF COMPILER2}
3886: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3887: {$ENDIF}
3888:
3889: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3890: begin
3891:   Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3892:   Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3893:   Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y: Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
3894:   Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3895:   Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3896:   Procedure BoxSetItem( List : TWinControl; Index : Integer)
3897:   Function BoxGetFirstSelection( List : TWinControl ) : Integer
3898:   Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer ) : Boolean
3899: end;
3900:
3901: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3902: begin
3903:   Const('MaxInitStrNum','LongInt' ( 9);
3904:   Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
: array of AnsiString; MaxSplit : Integer ) : Integer
3905:   Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
string; MaxSplit : Integer ) : Integer
3906:   Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3907:   Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3908:   Function JvStrStrip( S : string ) : string
3909:   Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3910:   Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar ) : AnsiString
3911:   Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3912:   Function StrEatWhiteSpace( const S : string) : string
3913:   Function HexToAscii( const S : AnsiString) : AnsiString
3914:   Function AsciiToHex( const S : AnsiString) : AnsiString
3915:   Function StripQuotes( const S1 : AnsiString) : AnsiString
3916:   Function ValidNumericLiteral( S1 : PAnsiChar ) : Boolean
3917:   Function ValidIntLiteral( S1 : PAnsiChar ) : Boolean
3918:   Function ValidHexLiteral( S1 : PAnsiChar ) : Boolean
3919:   Function HexPCharToInt( S1 : PAnsiChar ) : Integer

```

```

3920: Function ValidStringLiteral( S1 : PAnsiChar ) : Boolean
3921: Function StripPCharQuotes( S1 : PAnsiChar ) : AnsiString
3922: Function JvValidIdentifierAnsi( S1 : PAnsiChar ) : Boolean
3923: Function JvValidIdentifier( S1 : String ) : Boolean
3924: Function JvEndChar( X : AnsiChar ) : Boolean
3925: Procedure JvGetToken( S1, S2 : PAnsiChar )
3926: Function IsExpressionKeyword( S1 : PAnsiChar ) : Boolean
3927: Function IsKeyword( S1 : PAnsiChar ) : Boolean
3928: Function JvValidVarReference( S1 : PAnsiChar ) : Boolean
3929: Function GetParenthesis( S1, S2 : PAnsiChar ) : Boolean
3930: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar )
3931: Procedure JvEatWhitespaceChars( S1 : PAnsiChar );
3932: Procedure JvEatWhitespaceChars1( S1 : PWideChar );
3933: Function GetTokenCount : Integer
3934: Procedure ResetTokenCount
3935: end;
3936:
3937: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
3938: begin
3939:   SIRegister_TJvQueryParamsDialog(CL);
3940:   Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext ) : Boolean
3941:   end;
3942:
3943: ***** JvStrUtil / JvStrUtils;*****
3944: function FindNotBlankCharPos(const S: string): Integer;
3945: function AnsiChangeCase(const S: string): string;
3946: function GetWordOnPos(const S: string; const P: Integer): string;
3947: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3948: function Cmp(const S1, S2: string): Boolean;
3949: { Spaces returns string consists on N space chars }
3950: function Spaces(const N: Integer): string;
3951: { HasChar returns True, if char, Ch, contains in string, S }
3952: function HasChar(const Ch: Char; const S: string): Boolean;
3953: function HasAnyChar(const Chars: string; const S: string): Boolean;
3954: { SubStr returns substring from string, S, separated with Separator string }
3955: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3956: { SubStrEnd same to previous function but Index numerated from the end of string }
3957: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3958: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
3959: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3960: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3961: { GetXYByPos is same to previous function, but returns X position in line too }
3962: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3963: { AddSlash returns string with added slash char to Dir parameter, if needed }
3964: function AddSlash2(const Dir: TFileName): string;
3965: { AddPath returns FileName with Path, if FileName not contain any path }
3966: function AddPath(const FileName, Path: TFileName): TFileName;
3967: { ExePath returns ExtractFilePath(ParamStr(0)) }
3968: function ExePath: TFileName;
3969: function LoadTextFile(const FileName: TFileName): string;
3970: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3971: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3972: function ConcatSep(const S, S2, Separator: string): string;
3973: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3974: function FileEquMask(FileName, Mask: TFileName): Boolean;
3975: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3976:   Masks must be separated with comma (',' ) }
3977: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3978: function StringEndsWith(const Str, SubStr: string): Boolean;
3979: function ExtractFilePath2(const FileName: string): string;
3980: function StrToOem(const AnsiStr: string): string;
3981: { StrToOem translates a string from the Windows character set into the OEM character set. }
3982: function OemToAnsiStr(const OemStr: string): string;
3983: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
3984: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
3985: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
3986: function ReplaceStr(const S, Srch, Replace: string): string;
3987: { Returns string with every occurrence of Srch string replaced with Replace string. }
3988: function DelSpace(const S: string): string;
3989: { DelSpace return a string with all white spaces removed. }
3990: function DelChars(const S: string; Chr: Char): string;
3991: { DelChars return a string with all Chr characters removed. }
3992: function DelBSpace(const S: string): string;
3993: { DelBSpace trims leading spaces from the given string. }
3994: function DelESpace(const S: string): string;
3995: { DelESpace trims trailing spaces from the given string. }
3996: function DelRSpace(const S: string): string;
3997: { DelRSpace trims leading and trailing spaces from the given string. }
3998: function DelSpace1(const S: string): string;
3999: { DelSpace1 return a string with all non-single white spaces removed. }
4000: function Tab2Space(const S: string; Numb: Byte): string;
4001: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4002: function NPos(const C: string; S: string; N: Integer): Integer;
4003: { NPos searches for a N-th position of substring C in a given string. }
4004: function MakeStr(C: Char; N: Integer): string;
4005: function MS(C: Char; N: Integer): string;
4006: { MakeStr return a string of length N filled with character C. }
4007: function AddChar(C: Char; const S: string; N: Integer): string;
4008: { AddChar return a string left-padded to length N with characters C. }

```

```

4009: function AddCharR(C: Char; const S: string; N: Integer): string;
4010: { AddCharR return a string right-padded to length N with characters C. }
4011: function LeftStr(const S: string; N: Integer): string;
4012: { LeftStr return a string right-padded to length N with blanks. }
4013: function RightStr(const S: string; N: Integer): string;
4014: { RightStr return a string left-padded to length N with blanks. }
4015: function CenterStr(const S: string; Len: Integer): string;
4016: { CenterStr centers the characters in the string based upon the Len specified. }
4017: function CompStr(const S1, S2: string): Integer;
4018: { CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4019: function CompText(const S1, S2: string): Integer;
4020: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4021: function Copy2Symb(const S: string; Symb: Char): string;
4022: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4023: function Copy2SymbDel(var S: string; Symb: Char): string;
4024: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4025: function Copy2Space(const S: string): string;
4026: { Copy2Symb returns a substring of a string S from beginning to first white space. }
4027: function Copy2SpaceDel(var S: string): string;
4028: { Copy2SpaceDel returns a substring of a string S from beginning to first
4029:   white space and removes this substring from S. }
4030: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4031: { Returns string, with the first letter of each word in uppercase,
4032:   all other letters in lowercase. Words are delimited by WordDelims. }
4033: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4034: { WordCount given a set of word delimiters, returns number of words in S. }
4035: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4036: { Given a set of word delimiters, returns start position of N'th word in S. }
4037: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4038: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TCharSet; var Pos: Integer): string;
4039: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4040: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4041:   delimiters, return the N'th word in S. }
4042: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4043: { ExtractSubstr given set of word delimiters, returns the substring from S, that started from position Pos. }
4044: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4045: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4046: function QuotedString(const S: string; Quote: Char): string;
4047: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4048: function ExtractQuotedString(const S: string; Quote: Char): string;
4049: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4050:   and reduces pairs of Quote characters within the quoted string to a single character. }
4051: function FindPart(const HelpWilds, InputStr: string): Integer;
4052: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4053: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4054: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4055: function XorString(const Key, Src: ShortString): ShortString;
4056: function XorEncode(const Key, Source: string): string;
4057: function XorDecode(const Key, Source: string): string;
4058: { ** Command line routines ** }
4059: {$IFDEF COMPILER4_UP}
4060: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet; IgnoreCase: Boolean): Boolean;
4061: {$ENDIF}
4062: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4063: { ** Numeric string handling routines ** }
4064: function Numb2USA(const S: string): string;
4065: { Numb2USA converts numeric string S to USA-format. }
4066: function Dec2Hex(N: Longint; A: Byte): string;
4067: function D2H(N: Longint; A: Byte): string;
4068: { Dec2Hex converts the given value to a hexadecimal string representation
4069:   with the minimum number of digits (A) specified. }
4070: function Hex2Dec(const S: string): Longint;
4071: function H2D(const S: string): Longint;
4072: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4073: function Dec2Numb(N: Longint; A, B: Byte): string;
4074: { Dec2Numb converts the given value to a string representation with the
4075:   base equal to B and with the minimum number of digits (A) specified. }
4076: function Numb2Dec(S: string; B: Byte): Longint;
4077: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4078: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4079: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4080: function IntToRoman(Value: Longint): string;
4081: { IntToRoman converts the given value to a roman numeric string representation. }
4082: function RomanToInt(const S: string): Longint;
4083: { RomanToInt converts the given string to an integer value. If the string
4084:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4085: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4086: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4087: ***** JvFileUtil *****
4088: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4089: procedure CopyFileEx(const FileName, DestName: string; OverwriteReadOnly, ShellDialog: Boolean; ProgressControl:
  TControl);
4090: procedure MoveFile(const FileName, DestName: TFileName);
4091: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4092: {$IFDEF COMPILER4_UP}
4093: function GetFileSize(const FileName: string): Int64;
4094: {$ELSE}
4095: function GetFileSize(const FileName: string): Longint;
4096: {$ENDIF}

```

```

4097: function FileDateTime(const FileName: string): TDateTime;
4098: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4099: function DeleteFiles(const FileMask: string): Boolean;
4100: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4101: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4102: function NormalDir(const DirName: string): string;
4103: function RemoveBackSlash(const DirName: string): string;
4104: function ValidFileName(const FileName: string): Boolean;
4105: function DirExists(Name: string): Boolean;
4106: procedure ForceDirectories(Dir: string);
4107: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4108: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4109: {$IFDEF COMPILER4_UP}
4110: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4111: {$ENDIF}
4112: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4113: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4114: {$IFDEF COMPILER4_UP}
4115: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4116: {$ENDIF}
4117: function GetTempDir: string;
4118: function GetWindowsDir: string;
4119: function GetSystemDir: string;
4120: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4121: {$IFDEF WIN32}
4122: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4123: function ShortToLongFileName(const ShortName: string): string;
4124: function ShortToLongPath(const ShortName: string): string;
4125: function LongToShortFileName(const LongName: string): string;
4126: function LongToShortPath(const LongName: string): string;
4127: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4128: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4129: {$ENDIF WIN32}
4130: {$IFDEF COMPILER3_UP}
4131: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4132: {$ENDIF}
4133: function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext ) : TJvCalculatorForm
4134: function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode ) : TWinControl
4135: function CreatePopupCalculator( AOwner : TComponent ) : TWinControl
4136: procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4137:
4138: //*****procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4139: procedure VariantClear( var V : Variant)
4140: procedure VariantArrayRedim( var V : Variant; High : Integer)
4141: procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4142: procedure VariantCpy( const src : Variant; var dst : Variant)
4143: procedure VariantAdd( const src : Variant; var dst : Variant)
4144: procedure VariantSub( const src : Variant; var dst : Variant)
4145: procedure VariantMul( const src : Variant; var dst : Variant)
4146: procedure VariantDiv( const src : Variant; var dst : Variant)
4147: procedure VariantMod( const src : Variant; var dst : Variant)
4148: procedure VariantAnd( const src : Variant; var dst : Variant)
4149: procedure VariantOr( const src : Variant; var dst : Variant)
4150: procedure VariantXor( const src : Variant; var dst : Variant)
4151: procedure VariantShl( const src : Variant; var dst : Variant)
4152: procedure VariantShr( const src : Variant; var dst : Variant)
4153: function VariantAdd2( const V1 : Variant; const V2 : Variant ) : Variant
4154: function VariantSub2( const V1 : Variant; const V2 : Variant ) : Variant
4155: function VariantMul2( const V1 : Variant; const V2 : Variant ) : Variant
4156: function VariantDiv2( const V1 : Variant; const V2 : Variant ) : Variant
4157: function VariantMod2( const V1 : Variant; const V2 : Variant ) : Variant
4158: function VariantAnd2( const V1 : Variant; const V2 : Variant ) : Variant
4159: function VariantOr2( const V1 : Variant; const V2 : Variant ) : Variant
4160: function VariantXor2( const V1 : Variant; const V2 : Variant ) : Variant
4161: function VariantShl2( const V1 : Variant; const V2 : Variant ) : Variant
4162: function VariantShr2( const V1 : Variant; const V2 : Variant ) : Variant
4163: function VariantNot( const V1 : Variant ) : Variant
4164: function VariantNeg( const V1 : Variant ) : Variant
4165: function VariantGetElement( const V : Variant; i1 : integer ) : Variant;
4166: function VariantGetElement1( const V : Variant; i1, i2 : integer ) : Variant;
4167: function VariantGetElement2( const V : Variant; i1, i2, i3 : integer ) : Variant;
4168: function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer ) : Variant;
4169: function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer ) : Variant;
4170: procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4171: procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4172: procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4173: procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4174: procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4175: end;
4176:
4177: *****unit uPSI_JvgUtils;*****
4178: function IsEven(I: Integer): Boolean;
4179: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4180: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4181: procedure SwapInt(var I1, I2: Integer);
4182: function Spaces(Count: Integer): string;
4183: function DupStr(const Str: string; Count: Integer): string;
4184: function DupChar(C: Char; Count: Integer): string;
4185: procedure Msg(const AMsg: string);

```



```

4186: function RectW(R: TRect): Integer;
4187: function RectH(R: TRect): Integer;
4188: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4189: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4190: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4191: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4192:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4193: procedure DrawTextInRect(DC: HDC; R: TRect; const Text: string; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4194: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4195:   Style: TglTextStyle; ADelinedated, ASupress3D: Boolean; FontColor, DelinColor, HighlightColor, ShadowColor:
  TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4196: procedure DrawBox(DC: HDC; var R: TRect; Style: TglBoxStyle; BackgrColor: Longint; ATransparent: Boolean);
4197: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4198:   BevelInner, BevelOuter: TPanelBevel; Bold: Boolean; BackgrColor: Longint; ATransparent: Boolean): TRect;
4199: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient; PenStyle, PenWidth: Integer);
4200: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4201: procedure DrawBitmapExt(DC: HDC; { DC - background & result }
4202:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; {...X,Y _in_ rect}
4203:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4204:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4205: procedure CreateBitmapExt(DC: HDC; { DC - background & result }
4206:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; {...X,Y _in_ rect}
4207:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4208:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4209: procedure BringParentWindowToTop(Wnd: TWinControl);
4210: function GetParentForm(Control: TControl): TForm;
4211: procedure GetWindowImageFrom(Control: TWinControl; X, Y: Integer; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC)
4212: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4213: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4214: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4215: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4216: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4217: function CalcMathString(AExpression: string): Single;
4218: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4219: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4220: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4221: procedure TypeStringOnKeyboard(const S: string);
4222: function NextStringGridCell( Grid: TStringGrid ) : Boolean;
4223: procedure DrawTextExtAligned(Canvas: TCanvas; const
  Text: string; R: TRect; Alignment: TglAlignment; WordWrap: Boolean);
4224: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4225: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4226: function ComponentToString(Component: TComponent): string;
4227: procedure StringToComponent(Component: TComponent; const Value: string);
4228: function PlayWaveResource(const ResName: string): Boolean;
4229: function UserName: string;
4230: function ComputerName: string;
4231: function CreateIniFileName: string;
4232: function ExpandString(const Str: string; Len: Integer): string;
4233: function Transliterate(const Str: string; RusToLat: Boolean): string;
4234: function IsSmallFonts: Boolean;
4235: function SystemColorDepth: Integer;
4236: function GetFileTypeJ(const FileName: string): TglFileType;
4237: function GetFileType( hFile : THandle ) : DWORD';
4238: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4239: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4240:
4241: { *****Utility routines of unit classes}
4242: function LineStart(Buffer, BufPos: PChar): PChar
4243: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar; '+
4244:   'Strings: TStrings): Integer
4245: function TestStreamFormat( Stream : TStream ) : TStreamOriginalFormat
4246: procedure RegisterClass( AClass : TPersistentClass)
4247: procedure RegisterClasses( AClasses : array of TPersistentClass)
4248: procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4249: procedure UnRegisterClass( AClass : TPersistentClass)
4250: procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4251: procedure UnRegisterModuleClasses( Module : HMODULE)
4252: function FindGlobalComponent( const Name : string ) : TComponent
4253: function IsUniqueGlobalComponentName( const Name : string ) : Boolean
4254: function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass ) : Boolean
4255: function InitComponentRes( const ResName : string; Instance : TComponent ) : Boolean
4256: function ReadComponentRes( const ResName : string; Instance : TComponent ) : TComponent
4257: function ReadComponentResEx( HInstance : THandle; const ResName : string ) : TComponent
4258: function ReadComponentResFile( const FileName : string; Instance : TComponent ) : TComponent
4259: procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4260: procedure GlobalFixupReferences
4261: procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4262: procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4263: procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4264: procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4265: procedure RemoveFixups( Instance : TPersistent)
4266: function FindNestedComponent( Root : TComponent; const NamePath : string ) : TComponent
4267: procedure BeginGlobalLoading
4268: procedure NotifyGlobalLoading
4269: procedure EndGlobalLoading
4270: function GetUltimateOwner1( ACollection : TCollection ) : TPersistent;
4271: function GetUltimateOwner( APersistent : TPersistent ) : TPersistent;
4272: // AddTypes('TWndMethod', 'Procedure (var Message : TMessage)

```

```

4273: //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4274: Procedure FreeObjectInstance( ObjectInstance : Pointer)
4275: // Function AllocateHWnd( Method : TWndMethod) : HWND
4276: Procedure DeallocateHWnd( Wnd : HWND)
4277: Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4278: *****unit uPSI_SqlTimSt and DB;*****
4279: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4280: Function VarSQLTimeStampCreate3: Variant;
4281: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4282: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4283: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4284: Function VarSQLTimeStamp : TVarType
4285: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4286: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4287: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4288: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4289: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4290: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4291: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4292: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4293: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4294: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4295: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4296: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4297: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4298: //Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4299: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4300: Procedure DatabaseErrorFmt( const Message:WideString; const Args:array of const;Component:TComponent)
4301: Procedure DisposeMem( var Buffer, Size : Integer)
4302: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4303: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4304: Function VarTypeToDataType( VarType : Integer) : TFieldType
4305: *****unit JvStrings;*****
4306: {template functions}
4307: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4308: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4309: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4310: function RemoveMasterBlocks(const SourceStr: string): string;
4311: function RemoveFields(const SourceStr: string): string;
4312: {http functions}
4313: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4314: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4315: {set functions}
4316: procedure SplitSet(AText: string; AList: TStringList);
4317: function JoinSet(AList: TStringList): string;
4318: function FirstOfSet(const AText: string): string;
4319: function LastOfSet(const AText: string): string;
4320: function CountOfSet(const AText: string): Integer;
4321: function SetRotateRight(const AText: string): string;
4322: function SetRotateLeft(const AText: string): string;
4323: function SetPick(const AText: string; AIndex: Integer): string;
4324: function SetSort(const AText: string): string;
4325: function SetUnion(const Set1, Set2: string): string;
4326: function SetIntersect(const Set1, Set2: string): string;
4327: function SetExclude(const Set1, Set2: string): string;
4328: {replace any <,> etc by &lt; &gt;}
4329: function XMLSafe(const AText: string): string;
4330: {simple hash, Result can be used in Encrypt}
4331: function Hash(const AText: string): Integer;
4332: { Base64 encode and decode a string }
4333: function B64Encode(const S: AnsiString): AnsiString;
4334: function B64Decode(const S: AnsiString): AnsiString;
4335: {Basic encryption from a Borland Example}
4336: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4337: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4338: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4339: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4340: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4341: procedure CSVToTags(Src, Dst: TStringList);
4342: // converts a csv list to a tagged string list
4343: procedure TagsToCSV(Src, Dst: TStringList);
4344: // converts a tagged string list to a csv list
4345: // only fieldnames from the first record are scanned ib the other records
4346: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4347: {selects akey=avalue from Src and returns recordset in Dst}
4348: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4349: {filters Src for akey=avalue}
4350: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4351: {orders a tagged Src list by akey}
4352: function PosStr(const FindString, SourceString: string;
4353:   StartPos: Integer = 1): Integer;
4354: { PosStr searches the first occurrence of a substring FindString in a string
4355:   given by SourceString with case sensitivity (upper and lower case characters
4356:   are differed). This function returns the index value of the first character
4357:   of a specified substring from which it occurs in a given string starting with
4358:   StartPos character index. If a specified substring is not found Q_PosStr
4359:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4360: function PosStrLast(const FindString, SourceString: string): Integer;
4361: {finds the last occurrence}

```

```

4362: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4363: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4364: { PosText searches the first occurrence of a substring FindString in a string
4365:   given by SourceString without case sensitivity (upper and lower case characters are not differed). This
   function returns the index value of the first character of a specified substring from which it occurs in a
   given string starting with Start
4366: function PosTextLast(const FindString, SourceString: string): Integer;
4367: {finds the last occurrence}
4368: function NameValuesToXML(const AText: string): string;
4369: {$IFDEF MSWINDOWS}
4370: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4371: {$ENDIF MSWINDOWS}
4372: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4373: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4374: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4375: procedure SaveString(const AFile, AText: string);
4376: procedure SaveStringasFile(const AFile, AText: string);
4377: function LoadStringJ(const AFile: string): string;
4378: function LoadStringOfFile(const AFile: string): string;
4379: procedure SaveStringToFile(const AFile, AText: string);
4380: function LoadStringFromFile(const AFile: string): string;
4381: function HexToColor(const AText: string): TColor;
4382: function UppercaseHTMLTags(const AText: string): string;
4383: function LowercaseHTMLTags(const AText: string): string;
4384: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4385: function RelativePath(const ASrc, ADst: string): string;
4386: function GetToken(var Start: Integer; const SourceText: string): string;
4387: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4388: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4389: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4390: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4391: // parses the beginning of an attribute: space + alpha character
4392: function ParseAttribute(var Start: Integer; const SourceText: string; var AName, AValue: string): Boolean;
4393: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4394: procedure ParseAttributes(const SourceText: string; Attributes: TStringList);
4395: // parses all name=value attributes to the attributes TStringList
4396: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4397: // checks if a name="value" pair exists and returns any value
4398: function GetStrValue(const AText, AName, ADefault: string): string;
4399: // retrieves string value from a line like:
4400: // name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4401: // returns ADefault when not found
4402: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4403: // same for a color
4404: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4405: // same for an Integer
4406: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4407: // same for a float
4408: function GetBoolValue(const AText, AName: string): Boolean;
4409: // same for Boolean but without default
4410: function GetValue(const AText, AName: string): string;
4411: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4412: procedure SetValue(var AText: string; const AName, AValue: string);
4413: // sets a string value in a line
4414: procedure DeleteValue(var AText: string; const AName: string);
4415: // deletes a AName="value" pair from AText
4416: procedure GetNames(AText: string; AList: TStringList);
4417: // get a list of names from a string with name="value" pairs
4418: function GetHTMLColor(AColor: TColor): string;
4419: // converts a color value to the HTML hex value
4420: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4421: // finds a string backward case sensitive
4422: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4423: // finds a string backward case insensitive
4424: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4425:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4426: // finds a text range, e.g. <TD>...</TD> case sensitive
4427: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4428:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4429: // finds a text range, e.g. <TD>...</td> case insensitive
4430: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4431:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4432: // finds a text range backward, e.g. <TD>...</TD> case sensitive
4433: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4434:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4435: // finds a text range backward, e.g. <TD>...</td> case insensitive
4436: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4437:   var RangeEnd: Integer): Boolean;
4438: // finds a HTML or XML tag: <....>
4439: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4440:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4441: // finds the innertext between opening and closing tags
4442: function Easter(NYear: Integer): TDateTime;
4443: // returns the easter date of a year.
4444: function GetWeekNumber(Today: TDateTime): string;
4445: //gets a datecode. Returns year and weeknumber in format: YYWW
4446: function ParseNumber(const S: string): Integer;
4447: // parse number returns the last position, starting from 1
4448: function ParseDate(const S: string): Integer;

```

```

4449: // parse a SQL style data string from positions 1,
4450: // starts and ends with #
4451:
4452: *****unit JvJCLUtils;*****
4453:
4454: function VarIsInt(Value: Variant): Boolean;
4455: // VarIsInt returns VarIsOrdinal-[varBoolean]
4456: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4457: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4458: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4459: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4460: { GetWordOnPos returns Word from string, S, on the cursor position, P }
4461: function GetWordOnPos(const S: string; const P: Integer): string;
4462: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4463: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4464: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4465: { GetWordOnPosEx working like GetWordOnPos function, but
4466:   also returns Word position in iBeg, iEnd variables }
4467: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4468: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4469: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4470: function GetNextWordPosExW(const Text: WideString; StartIndex: Integer; var iBeg, iEnd: Integer): WideString;
4471: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4472: { GetEndPosCaret returns the caret position of the last char. For the position
4473:   after the last char of Text you must add 1 to the returned X value. }
4474: procedure GetEndPosCaretW(const Text: WideString; CaretX, CaretY: Integer; var X, Y: Integer);
4475: { GetEndPosCaret returns the caret position of the last char. For the position
4476:   after the last char of Text you must add 1 to the returned X value. }
4477: { SubStrBySeparator returns substring from string, S, separated with Separator string }
4478: function SubStrBySeparator(const S: string; const Index: Integer; const
  Separator: string; StartIndex: Integer = 1): string;
4479: function SubStrBySeparatorW(const S: WideString; const Index: Integer; const
  Separator: WideString; StartIndex: Integer = 1): WideString;
4480: { SubStrEnd same to previous function but Index numerated from the end of string }
4481: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4482: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4483: function SubWord(P: PChar; var P2: PChar): string;
4484: function CurrencyByWord(Value: Currency): string;
4485: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4486: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4487: { GetXYByPos is same as GetLineByPos, but returns X position in line as well }
4488: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4489: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4490: { ReplaceString searches for all substrings, OldPattern,
4491:   in a string, S, and replaces them with NewPattern }
4492: function ReplaceString(S: string; const OldPattern, NewPattern: string; StartIndex: Integer = 1): string;
4493: function ReplaceStringW(S: WideString; const OldPattern, NewPattern:
  WideString; StartIndex: Integer = 1): WideString;
4494: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4495: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
  SUPPORTS_INLINE}
4496: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
4497: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
  SUPPORTS_INLINE}
4498:
4499: { Next 4 function for russian chars transliterating.
4500:   This functions are needed because Oem2Ansi and AnsiOem functions sometimes suck }
4501: procedure Dos2Win(var S: AnsiString);
4502: procedure Win2Dos(var S: AnsiString);
4503: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4504: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4505: function Win2Koi(const S: AnsiString): AnsiString;
4506: { FillString fills the string Buffer with Count Chars }
4507: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4508: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4509: { MoveString copies Count Chars from Source to Dest }
4510: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
  inline; {$ENDIF SUPPORTS_INLINE} overload;
4511: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
  DstStartIdx: Integer; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4512: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4513: procedure FillWideChar(var Buffer: string; Count: Integer; const Value: WideChar);
4514: { MoveWideChar copies Count WideChars from Source to Dest }
4515: procedure MoveWideChar(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
  inline; {$ENDIF SUPPORTS_INLINE}
4516: { FillNativeChar fills Buffer with Count NativeChars }
4517: procedure FillNativeChar(var Buffer: string; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
  SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4518: { MoveNativeChar copies Count WideChars from Source to Dest }
4519: procedure MoveNativeChar(const Source: string; var Dest: string; Count: Integer); // D2009 internal error {$IFDEF
  SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4520: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4521: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4522: { Spaces returns string consists on N space chars }
4523: function Spaces(const N: Integer): string;
4524: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4525: function AddSpaces(const S: string; const N: Integer): string;
4526: function SpacesW(const N: Integer): WideString;
4527: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4528:

```



```

4529: { function LastDateRUS for russian users only }
4530: { returns date relative to current date: 'âââ âîý îâçââ' }
4531: function LastDateRUS(const Dat: TDateTime): string;
4532: { CurrencyToStr format Currency, Cur, using ffCurrency float format }
4533: function CurrencyToStr(const Cur: Currency): string;
4534: { HasChar returns True, if Char, Ch, contains in string, S }
4535: function HasChar(const Ch: Char; const S: string): Boolean;
4536: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4537: function HasAnyChar(const Chars: string; const S: string): Boolean;
4538: {$IFDEF COMPILER12_UP}
4539: function CharInSet(const Ch: AnsiChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4540: {$ENDIF ~COMPILER12_UP}
4541: function CharInSetW(const Ch: WideChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF
SUPPORTS_INLINE}
4542: function CountOfChar(const Ch: Char; const S: string): Integer;
4543: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
SUPPORTS_INLINE}
4544: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4545: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4546: function StrPosW(S, SubStr: PWideChar): PWideChar;
4547: function StrLenW(S: PWideChar): Integer;
4548: function TrimW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4549: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
SUPPORTS_INLINE}
4550: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4551: TPixelFormat, '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4552: TMappingMethod, '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4553: function GetBitmapPixelFormat( Bitmap : TBitmap ) : TPixelFormat
4554: function GetPaletteBitmapFormat( Bitmap : TBitmap ) : TPixelFormat
4555: procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4556: function BitmapToMemoryStream(Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod): TMemoryStream;
4557: procedure GrayscaleBitmap( Bitmap : TBitmap)
4558: function BitmapToMemory( Bitmap : TBitmap; Colors : Integer ) : TStream
4559: procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4560: function ScreenPixelFormat : TPixelFormat
4561: function ScreenColorCount : Integer
4562: procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4563: function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean ) : TPoint
4564: // SIRegister_TJvGradient(CL);
4565:
4566: {***** files routines}
4567: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4568: const
4569: {$IFDEF MSWINDOWS}
4570: DefaultCaseSensitivity = False;
4571: {$ENDIF MSWINDOWS}
4572: {$IFDEF UNIX}
4573: DefaultCaseSensitivity = True;
4574: {$ENDIF UNIX}
4575: { GenTempFileName returns temporary file name on
4576: drive, there FileName is placed }
4577: function GenTempFileName(FileName: string): string;
4578: { GenTempFileNameExt same to previous function, but
4579: returning filename has given extension, FileExt }
4580: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4581: { ClearDir clears folder Dir }
4582: function ClearDir(const Dir: string): Boolean;
4583: { DeleteDir clears and than delete folder Dir }
4584: function DeleteDir(const Dir: string): Boolean;
4585: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4586: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4587: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4588: Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4589: function FileEquMasks(FileName, Masks: TFileName;
4590: CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4591: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4592: {$IFDEF MSWINDOWS}
4593: { LZFileExpand expand file, FileSource,
4594: into FileDest. Given file must be compressed, using MS Compress program }
4595: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4596: {$ENDIF MSWINDOWS}
4597: { FileGetInfo fills SearchRec record for specified file attributes}
4598: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4599: { HasSubFolder returns True, if folder APath contains other folders }
4600: function HasSubFolder(APath: TFileName): Boolean;
4601: { IsEmptyFolder returns True, if there are no files or
4602: folders in given folder, APath}
4603: function IsEmptyFolder(APath: TFileName): Boolean;
4604: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4605: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4606: { AddPath returns FileName with Path, if FileName not contain any path }
4607: function AddPath(const FileName, Path: TFileName): TFileName;
4608: function AddPaths(const PathList, Path: string): string;
4609: function ParentPath(const Path: TFileName): TFileName;
4610: function FindInPath(const FileName, PathList: string): TFileName;
4611: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4612: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4613: { HasParam returns True, if program running with specified parameter, Param }
4614: function HasParam(const Param: string): Boolean;

```

```

4615: function HasSwitch(const Param: string): Boolean;
4616: function Switch(const Param: string): string;
4617: { ExePath returns ExtractFilePath(ParamStr(0)) }
4618: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4619: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4620: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4621: procedure FileTimeToDosDateTimeDword(const FT: TFileTime; out Dft: DWORD);
4622: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4623: {**** Graphic routines }
4624: { IstTFontSelected returns True, if True Type font is selected in specified device context }
4625: function IstTFontSelected(const DC: HDC): Boolean;
4626: function KeyPressed(VK: Integer): Boolean;
4627: function isKeyPressed: boolean; //true if key on memo2 (shell output) is pressed
4628: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4629: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4630: {**** Color routines }
4631: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4632: function RGBToBGR(Value: Cardinal): Cardinal;
4633: //function ColorToPrettyName(Value: TColor): string;
4634: //function PrettyNameToColor(const Value: string): TColor;
4635: {**** other routines }
4636: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4637: function IntPower(Base, Exponent: Integer): Integer;
4638: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4639: function StrToBool(const S: string): Boolean;
4640: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4641: function VarToInt(V: Variant): Integer;
4642: function VarToFloat(V: Variant): Double;
4643: { following functions are not documented because they not work properly sometimes, so do not use them }
4644: // (rom) ReplaceStrings1, GetSubStr removed
4645: function GetLongFileName(const FileName: string): string;
4646: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4647: function GetParameter: string;
4648: function GetComputerID: string;
4649: function GetComputerName: string;
4650: {**** string routines }
4651: { ReplaceAllStrings searches for all substrings, Words,
4652:   in a string, S, and replaces them with Phrases with the same Index. }
4653: function ReplaceAllStrings(const S: string; Words, Phrases: TStrings): string;
4654: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4655:   in the list, Words, and if founds, replaces this Word with string from another list, Phrases, with the
   same Index, and then update NewSelStart variable }
4656: function ReplaceStrings(const S: string; PosBeg, Len: Integer; Words, Phrases: TStrings; var NewSelStart: Integer): string;
4657: { CountOfLines calculates the lines count in a string, S,
4658:   each line must be separated from another with CrLf sequence }
4659: function CountOfLines(const S: string): Integer;
4660: { DeleteLines deletes all lines from strings which in the words, words.
4661:   The word of will be deleted from strings. }
4662: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4663: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4664:   Lines contained only spaces also deletes. }
4665: procedure DeleteEmptyLines(Ss: TStrings);
4666: { SQLAddWhere addes or modifies existing where-statement, where,
4667:   to the strings, SQL. Note: If strings SQL already contains where-statement,
4668:   it must be started on the begining of any line }
4669: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4670: {**** files routines - }
4671: {$IFDEF MSWINDOWS}
4672: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4673:   Resource can be compressed using MS Compress program }
4674: function ResSaveToFile(const Typ, Name: string; const Compressed: Boolean; const FileName: string): Boolean;
4675: function ResSaveToFileEx(Instance: HINST; Typ, Name: PChar; const Compressed: Boolean; const FileName: string):
   Boolean;
4676: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4677: {$ENDIF MSWINDOWS}
4678: { IniReadSection read section, Section, from ini-file,
4679:   IniFileName, into strings, Ss. This function reads ALL strings from specified section.
4680:   Note: TIniFile.ReadSection function reads only strings with '=' symbol. }
4681: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4682: { LoadTextFile load text file, FileName, into string }
4683: function LoadTextFile(const FileName: TFileName): string;
4684: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4685: { ReadFolder reads files list from disk folder, Folder, that are equal to mask, Mask, into strings, FileList }
4686: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4687: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4688: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4689: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4690: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4691: function RATextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: string; const CalcHeight: Boolean): Integer;
4692: { RATextCalcHeight calculate needed height for
4693:   correct output, using RATextOut or RATextOutEx functions }
4694: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4695: { Cinema draws some visual effect }
4696: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4697: { Roughed fills rect with special 3D pattern }
4698: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4699: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
   and height, AWidth, AHeight and placed on a specified Index, Index in the source bitmap }
4700: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;

```

```

4701: { TextWidth calculate text width for writing using standard desktop font }
4702: function TextWidth(const AStr: string): Integer;
4703: { TextHeight calculate text height for writing using standard desktop font }
4704: function TextHeight(const AStr: string): Integer;
4705: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4706: procedure Error(const Msg: string);
4707: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4708:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4709: {example Text parameter: 'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4710: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4711:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4712: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4713:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4714: function ItemHtPlain(const Text: string): string;
4715: { ClearList - clears list of TObject }
4716: procedure ClearList(List: TList);
4717: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
4718: procedure ClipboardToMemStream(MemStream: TMemoryStream; const Format: Word);
4719: { RTTI support }
4720: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4721: function GetPropStr(Obj: TObject; const PropName: string): string;
4722: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4723: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4724: procedure PrepareIniSection(Ss: TStringList);
4725: { following functions are not documented because they are don't work properly, so don't use them }
4726: // (rom) from JvBandWindows to make it obsolete
4727: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4728: // (rom) from JvBandUtils to make it obsolete
4729: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4730: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4731: function CreateIconFromClipboard: TIcon;
4732: { begin JvIconClipboardUtils } { Icon clipboard routines }
4733: function CF_ICON: Word;
4734: procedure AssignClipboardIcon(Icon: TIcon);
4735: { Real-size icons support routines (32-bit only) }
4736: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4737: function CreateRealSizeIcon(Icon: TIcon): HICON;
4738: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4739: {end JvIconClipboardUtils }
4740: function CreateScreenCompatibleDC: HDC;
4741: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
SUPPORTS_INLINE} inline; {$ENDIF}
4742: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4743: { begin JvRLE } // (rom) changed API for inclusion in JCL
4744: procedure RleCompressTo(InStream, OutStream: TStream);
4745: procedure RleDecompressTo(InStream, OutStream: TStream);
4746: procedure RleCompress(Stream: TStream);
4747: procedure RleDecompress(Stream: TStream);
4748: { end JvRLE } { begin JvDateUtil }
4749: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4750: function IsLeapYear(AYear: Integer): Boolean;
4751: function DaysInAMonth(const AYear, AMonth: Word): Word;
4752: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4753: function FirstDayOfPrevMonth: TDateTime;
4754: function LastDayOfPrevMonth: TDateTime;
4755: function FirstDayOfNextMonth: TDateTime;
4756: function ExtractDay(ADate: TDateTime): Word;
4757: function ExtractMonth(ADate: TDateTime): Word;
4758: function ExtractYear(ADate: TDateTime): Word;
4759: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4760: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4761: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4762: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4763: function ValidDate(ADate: TDateTime): Boolean;
4764: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4765: function MonthsBetween(Date1, Date2: TDateTime): Double;
4766: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4767: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4768: function DaysBetween(Date1, Date2: TDateTime): Longint;
4769: { The same as previous but if Date2 < Date1 result = 0 }
4770: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4771: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4772: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4773: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4774: function IncMsec(ATime: TDateTime; Delta: Integer): TDateTime;
4775: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
4776: { String to date conversions }
4777: function GetDateOrder(const DateFormat: string): TDateOrder;
4778: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4779: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4780: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4781: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4782: //function DefDateFormat(AFourDigitYear: Boolean): string;
4783: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4784: function FormatLongDate(Value: TDateTime): string;
4785: function FormatLongDateTime(Value: TDateTime): string;
4786: { end JvDateUtil }
4787: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4788: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;

```

```

4789: { begin JvStrUtils } { ** Common string handling routines ** }
4790: {$IFDEF UNIX}
4791: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4792:   const ToCode, FromCode: AnsiString): Boolean;
4793: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4794: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4795: function OemStrToAnsi(const S: AnsiString): AnsiString;
4796: function AnsiStrToOem(const S: AnsiString): AnsiString;
4797: {$ENDIF UNIX}
4798: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4799: { StrToOem translates a string from the Windows character set into the OEM character set. }
4800: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4801: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4802: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4803: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4804: function ReplaceStr(const S, Srch, Replace: string): string;
4805: { Returns string with every occurrence of Srch string replaced with Replace string. }
4806: function DelSpace(const S: string): string;
4807: { DelSpace return a string with all white spaces removed. }
4808: function DelChars(const S: string; Chr: Char): string;
4809: { DelChars return a string with all Chr characters removed. }
4810: function DelBSpace(const S: string): string;
4811: { DelBSpace trims leading spaces from the given string. }
4812: function DelEspace(const S: string): string;
4813: { DelEspace trims trailing spaces from the given string. }
4814: function DelRSpace(const S: string): string;
4815: { DelRSpace trims leading and trailing spaces from the given string. }
4816: function DelSpace1(const S: string): string;
4817: { DelSpace1 return a string with all non-single white spaces removed. }
4818: function Tab2Space(const S: string; Numb: Byte): string;
4819: { Tab2Space converts any tabulation character in the given string to the
4820:   Numb spaces characters. }
4821: function NPos(const C: string; S: string; N: Integer): Integer;
4822: { NPos searches for a N-th position of substring C in a given string. }
4823: function MakeStr(C: Char; N: Integer): string; overload;
4824: {$IFDEF COMPILER12_UP}
4825: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4826: {$ENDIF !COMPILER12_UP}
4827: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4828: { MakeStr return a string of length N filled with character C. }
4829: function AddChar(C: Char; const S: string; N: Integer): string;
4830: { AddChar return a string left-padded to length N with characters C. }
4831: function AddCharR(C: Char; const S: string; N: Integer): string;
4832: { AddCharR return a string right-padded to length N with characters C. }
4833: function LeftStr(const S: string; N: Integer): string;
4834: { LeftStr return a string right-padded to length N with blanks. }
4835: function RightStr(const S: string; N: Integer): string;
4836: { RightStr return a string left-padded to length N with blanks. }
4837: function CenterStr(const S: string; Len: Integer): string;
4838: { CenterStr centers the characters in the string based upon the Len specified. }
4839: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4840: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4841:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4842: function CompText(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4843: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4844: function Copy2Symb(const S: string; Symb: Char): string;
4845: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4846: function Copy2SymbDel(var S: string; Symb: Char): string;
4847: { Copy2SymbDel returns a substring of a string S from beginning to first
4848:   character Symb and removes this substring from S. }
4849: function Copy2Space(const S: string): string;
4850: { Copy2Symb returns a substring of a string S from beginning to first white space. }
4851: function Copy2SpaceDel(var S: string): string;
4852: { Copy2SpaceDel returns a substring of a string S from beginning to first
4853:   white space and removes this substring from S. }
4854: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4855: { Returns string, with the first letter of each word in uppercase,
4856:   all other letters in lowercase. Words are delimited by WordDelims. }
4857: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4858: { WordCount given a set of word delimiters, returns number of words in S. }
4859: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4860: { Given a set of word delimiters, returns start position of N'th word in S. }
4861: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4862: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TSysCharSet; var Pos: Integer): string;
4863: function ExtractDelimited(N: Integer; const S: string; const Delims: TSysCharSet): string;
4864: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4865:   delimiters, return the N'th word in S. }
4866: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TSysCharSet): string;
4867: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4868:   that started from position Pos. }
4869: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4870: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4871: function QuotedString(const S: string; Quote: Char): string;
4872: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4873: function ExtractQuotedString(const S: string; Quote: Char): string;
4874: { ExtractQuotedString removes the Quote characters from the beginning and
4875:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character. }
4876: function FindPart(const HelpWilds, InputStr: string): Integer;
4877: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }

```



```

4878: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4879: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4880: function XorString(const Key, Src: ShortString): ShortString;
4881: function XorEncode(const Key, Source: string): string;
4882: function XorDecode(const Key, Source: string): string;
4883: { ** Command line routines ** }
4884: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4885: { ** Numeric string handling routines ** }
4886: function Numb2USA(const S: string): string;
4887: { Numb2USA converts numeric string S to USA-format. }
4888: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4889: { Dec2Hex converts the given value to a hexadecimal string representation
4890:   with the minimum number of digits (A) specified. }
4891: function Hex2Dec(const S: string): Longint;
4892: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4893: function Dec2Numb(N: Int64; A, B: Byte): string;
4894: { Dec2Numb converts the given value to a string representation with the
4895:   base equal to B and with the minimum number of digits (A) specified. }
4896: function Numb2Dec(S: string; B: Byte): Int64;
4897: { Numb2Dec converts the given B-based numeric string to the corresponding
4898:   integer value. }
4899: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4900: { IntToBin converts the given value to a binary string representation
4901:   with the minimum number of digits specified. }
4902: function IntToRoman(Value: Longint): string;
4903: { IntToRoman converts the given value to a roman numeric string representation. }
4904: function RomanToInt(const S: string): Longint;
4905: { RomanToInt converts the given string to an integer value. If the string
4906:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4907: function FindNotBlankCharPos(const S: string): Integer;
4908: function FindNotBlankCharPosW(const S: WideString): Integer;
4909: function AnsiChangeCase(const S: string): string;
4910: function WideChangeCase(const S: string): string;
4911: function StartsText(const SubStr, S: string): Boolean;
4912: function EndText(const SubStr, S: string): Boolean;
4913: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4914: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4915: {end JvStrUtils}
4916: {$IFDEF UNIX}
4917: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4918: {$ENDIF UNIX}
4919: { begin JvFileUtil }
4920: function FileDateTime(const FileName: string): TDateTime;
4921: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4922: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4923: function NormalDir(const DirName: string): string;
4924: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4925: function ValidFileName(const FileName: string): Boolean;
4926: {$IFDEF MSWINDOWS}
4927: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4928: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4929: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4930: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4931: {$ENDIF MSWINDOWS}
4932: function GetWindowsDir: string;
4933: function GetSystemDir: string;
4934: function ShortToLongFileName(const ShortName: string): string;
4935: function LongToShortFileName(const LongName: string): string;
4936: function ShortToLongPath(const ShortName: string): string;
4937: function LongToShortPath(const LongName: string): string;
4938: {$IFDEF MSWINDOWS}
4939: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4940: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4941: {$ENDIF MSWINDOWS}
4942: { end JvFileUtil }
4943: // Works like PtInRect but includes all edges in comparison
4944: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4945: // Works like PtInRect but excludes all edges from comparison
4946: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4947: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4948: function IsFourDigitYear: Boolean;
4949: { moved from JvJCLUtils }
4950: //Open an object with the shell (url or something like that)
4951: function OpenObject(const Value: string): Boolean; overload;
4952: function OpenObject(Value: PChar): Boolean; overload;
4953: {$IFDEF MSWINDOWS}
4954: //Raise the last Exception
4955: procedure RaiseLastWin32; overload;
4956: procedure RaiseLastWin32(const Text: string); overload;
4957: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
4958:   significant 32 bits of a file's binary version number. Typically, this includes the major and minor
4959:   version placed together in one 32-bit Integer. I
4960: function GetFileVersion(const AFileName: string): Cardinal;
4961: {$EXTERNALSYM GetFileVersion}
4962: //Get version of Shell.dll
4963: function GetShellVersion: Cardinal;
4964: {$EXTERNALSYM GetShellVersion}
4965: // CD functions on HW
4966: procedure OpenCdDrive;

```

```

4965: procedure CloseCdDrive;
4966: // returns True if Drive is accessible
4967: function DiskInDrive(Drive: Char): Boolean;
4968: {$SENDIF MSWINDOWS}
4969: //Same as linux function ;)
4970: procedure PError(const Text: string);
4971: // execute a program without waiting
4972: procedure Exec(const FileName, Parameters, Directory: string);
4973: // execute a program and wait for it to finish
4974: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
4975: // returns True if this is the first instance of the program that is running
4976: function FirstInstance(const ATitle: string): Boolean;
4977: // restores a window based on it's classname and Caption. Either can be left empty
4978: // to widen the search
4979: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
4980: // manipulate the traybar and start button
4981: procedure HideTraybar;
4982: procedure ShowTraybar;
4983: procedure ShowStartButton(Visible: Boolean = True);
4984: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
4985: procedure MonitorOn;
4986: procedure MonitorOff;
4987: procedure LowPower;
4988: // send a key to the window named AppName
4989: function SendKey(const AppName: string; Key: Char): Boolean;
4990: {$IFDEF MSWINDOWS}
4991: // returns a list of all win currently visible, the Objects property is filled with their window handle
4992: procedure GetVisibleWindows(List: TStringList);
4993: function GetVisibleWindowsF( List : TStringList):TStringList;
4994: // associates an extension to a specific program
4995: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
4996: procedure AddToRecentDocs(const FileName: string);
4997: function GetRecentDocs: TStringList;
4998: {$SENDIF MSWINDOWS}
4999: function CharIsMoney(const Ch: Char): Boolean;
5000: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5001: function IntToExtended(I: Integer): Extended;
5002: { GetChangedText works out the new text given the current cursor pos & the key pressed
5003:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5004: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5005: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5006: //function StrIsInteger(const S: string): Boolean;
5007: function StrIsFloatMoney(const Ps: string): Boolean;
5008: function StrIsDateTime(const Ps: string): Boolean;
5009: function PreformatDateString(Ps: string): string;
5010: function BooleanToInteger(const B: Boolean): Integer;
5011: function StringToBoolean(const Ps: string): Boolean;
5012: function SafeStrToDateTime(const Ps: string): TDateTime;
5013: function SafeStrToDate(const Ps: string): TDateTime;
5014: function SafeStrToTime(const Ps: string): TDateTime;
5015: function StrDelete(const psSub, psMain: string): string;
5016: { returns the fractional value of pcValue }
5017: function TimeOnly(pcValue: TDateTime): TTime;
5018: { returns the integral value of pcValue }
5019: function DateOnly(pcValue: TDateTime): TDate;
5020: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5021: const { TDateTime value used to signify Null value }
5022:   NullEquivalentDate: TDateTime = 0.0;
5023: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5024: // Replacement for Win32Check to avoid platform specific warnings in D6
5025: function OSCheck(RetVal: Boolean): Boolean;
5026: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5027:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5028:   not be forced to use FileCtrl unnecessarily }
5029: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5030: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5031: { MinimizeString truncates long string, S, and appends...'symbols',if Length of S is more than MaxLen }
5032: function MinimizeString(const S: string; const MaxLen: Integer): string;
5033: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
SW_SHOWDEFAULT);
5034: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion function and calls it, returning major and
minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't be
found.}
5035: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5036: {$SENDIF MSWINDOWS}
5037: procedure ResourceNotFound(ResID: PChar);
5038: function EmptyRect: TRect;
5039: function RectWidth(R: TRect): Integer;
5040: function RectHeight(R: TRect): Integer;
5041: function CompareRect(const R1, R2: TRect): Boolean;
5042: procedure RectNormalize(var R: TRect);
5043: function RectIsSquare(const R: TRect): Boolean;
5044: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5045: //If AMaxSize = -1 ,then auto calc Square's max size
5046: {$IFDEF MSWINDOWS}
5047: procedure FreeUnusedOle;
5048: function GetWindowsVersion: string;
5049: function LoadDLL(const LibName: string): THandle;
5050: function RegisterServer(const ModuleName: string): Boolean;

```

```

5051: function UnregisterServer(const ModuleName: string): Boolean;
5052: { $ENDIF MSWINDOWS }
5053: { String routines }
5054: function GetEnvVar(const VarName: string): string;
5055: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5056: function StringToPChar(var S: string): PChar;
5057: function StrPAlloc(const S: string): PChar;
5058: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5059: function DropT(const S: string): string;
5060: { Memory routines }
5061: function AllocMemo(Size: Longint): Pointer;
5062: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5063: procedure FreeMemo(var fpBlock: Pointer);
5064: function GetMemoSize(fpBlock: Pointer): Longint;
5065: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5066: { Manipulate huge pointers routines }
5067: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5068: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5069: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5070: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5071: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5072: function WindowClassName(Wnd: THandle): string;
5073: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5074: procedure ActivateWindow(Wnd: THandle);
5075: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5076: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5077: { SetWindowTop put window to top without recreating window }
5078: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5079: procedure CenterWindow(Wnd: THandle);
5080: function MakeVariant(const Values: array of Variant): Variant;
5081: { Convert dialog units to pixels and backwards }
5082: { $IFDEF MSWINDOWS }
5083: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5084: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5085: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5086: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5087: { $ENDIF MSWINDOWS }
5088: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5089: { $IFDEF BCB }
5090: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): THandle;
5091: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
5092: { $ELSE }
5093: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5094: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5095: { $ENDIF BCB }
5096: { $IFDEF MSWINDOWS }
5097: { BrowseForFolderNative displays Browse For Folder dialog }
5098: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5099: { $ENDIF MSWINDOWS }
5100: procedure AntiAlias(Clip: TBitmap);
5101: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin, XFinal, YFinal: Integer);
5102: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5103:   ABitmap: TBitmap; const SourceRect: TRect);
5104: function IsTrueType(const FontName: string): Boolean;
5105: // Removes all non-numeric characters from AValue and returns the resulting string
5106: function TextToValText(const AValue: string): string;
5107: function ExecRegExpr(const ARegExpr, AInputStr: RegExprString): boolean;
5108: procedure SplitRegExpr(const ARegExpr, AInputStr: RegExprString; APieces: TStrings);
5109: function ReplaceRegExpr(const ARegExpr, AInputStr,
  AReplaceStr: RegExprString; AUseSubstitution: boolean): RegExprString;
5110: function QuoteRegExprMetaChars(const AStr: RegExprString): RegExprString;
5111: function RegExprSubExpressions(const ARegExpr: string; ASubExprs: TStrings; AExtendedSyntax: boolean):
5112:
5113: *****unit uPSI_JvTFUtils;
5114: function JExtractYear( ADate: TDateTime): Word;
5115: function JExtractMonth( ADate: TDateTime): Word;
5116: function JExtractDay( ADate: TDateTime): Word;
5117: function ExtractHours( ATime: TDateTime): Word;
5118: function ExtractMins( ATime: TDateTime): Word;
5119: function ExtractSecs( ATime: TDateTime): Word;
5120: function ExtractMSecs( ATime: TDateTime): Word;
5121: function FirstOfMonth( ADate: TDateTime): TDateTime;
5122: function GetDayOfNthDOW( Year, Month, DOW, N: Word): Word;
5123: function GetWeeksInMonth( Year, Month: Word; StartOfWeek: Integer): Word;
5124: procedure IncBorlDOW( var BorlDOW: Integer; N: Integer);
5125: procedure IncDOW( var DOW: TTFDayOfWeek; N: Integer);
5126: procedure IncDays( var ADate: TDateTime; N: Integer);
5127: procedure IncWeeks( var ADate: TDateTime; N: Integer);
5128: procedure IncMonths( var ADate: TDateTime; N: Integer);
5129: procedure IncYears( var ADate: TDateTime; N: Integer);
5130: function EndOfMonth( ADate: TDateTime): TDateTime;
5131: function IsFirstOfMonth( ADate: TDateTime): Boolean;
5132: function IsEndOfMonth( ADate: TDateTime): Boolean;
5133: procedure EnsureMonth( Month: Word);
5134: procedure EnsureDOW( DOW: Word);
5135: function EqualDates( D1, D2: TDateTime): Boolean;
5136: function Lesser( N1, N2: Integer): Integer;
5137: function Greater( N1, N2: Integer): Integer;
5138: function GetDivLength( TotalLength, DivCount, DivNum: Integer): Integer;

```

```

5139: Function GetDivNum( TotalLength, DivCount, X : Integer ) : Integer
5140: Function GetDivStart( TotalLength, DivCount, DivNum : Integer ) : Integer
5141: Function DOWToBorl( ADOW : TTFDayOfWeek ) : Integer
5142: Function BorlToDOW( BorlDOW : Integer ) : TTFDayOfWeek
5143: Function DateToDOW( ADate : TDateTime ) : TTFDayOfWeek
5144: Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5145: Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5146: Function JRectWidth( ARect : TRect ) : Integer
5147: Function JRectHeight( ARect : TRect ) : Integer
5148: Function JEmptyRect : TRect
5149: Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5150:
5151: procedure SIRegister_MSUtils(CL: TPSPascalCompiler);
5152: begin
5153: Procedure HideTaskBarButton( hWindow : HWND)
5154: Function msLoadStr( ID : Integer ) : String
5155: Function msFormat( fmt : String; params : array of const) : String
5156: Function msFileExists( const FileName : String) : Boolean
5157: Function msIntToStr( Int : Int64 ) : String
5158: Function msStrPas( const Str : PChar ) : String
5159: Function msRenameFile( const OldName, NewName : String) : Boolean
5160: Function CutFileName( s : String) : String
5161: Function GetVersionInfo( var VersionString : String) : DWORD
5162: Function FormatTime( t : Cardinal ) : String
5163: Function msCreateDir( const Dir : string) : Boolean
5164: Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5165: Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword ) : DWORD
5166: Function msStrLen( Str : PChar ) : Integer
5167: Function msDirectoryExists( const Directory : String) : Boolean
5168: Function GetFolder( hWnd : HWND; RootDir : Integer; Caption : String) : String
5169: Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5170: Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
5171: Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5172: Function GetTextFromFile( Filename : String) : string
5173: Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5174: Function msStrToIntDef( const s : String; const i : Integer) : Integer
5175: Function msStrToInt( s : String) : Integer
5176: Function GetItemText( hDlg : THandle; ID : DWORD) : String
5177: end;
5178:
5179: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5180: begin
5181: //TDynFloatArray', 'array of Extended
5182: TDynLWordArray', 'array of LongWord
5183: TDynLIntArray', 'array of LongInt
5184: TDynFloatMatrix', 'array of TDynFloatArray
5185: TDynLWordMatrix', 'array of TDynLWordArray
5186: TDynLIntMatrix', 'array of TDynLIntArray
5187: Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5188: Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5189: Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5190: Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5191: Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5192: Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5193: Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5194: Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5195: Function DotProduct( const X, Y : TDynFloatArray) : Extended
5196: Function MNorm( const X : TDynFloatArray) : Extended
5197: Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5198: Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5199: Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5200: Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5201: Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5202: Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5203: Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5204: Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5205: Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5206: Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5207: Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5208: Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5209: Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5210: end;
5211:
5212: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5213: begin
5214: 'ESBminSingle','Single').setExtended( 1.5e-45);
5215: 'ESBmaxSingle','Single').setExtended( 3.4e+38);
5216: 'ESBminDouble','Double').setExtended( 5.0e-324);
5217: 'ESBmaxDouble','Double').setExtended( 1.7e+308);
5218: 'ESBminExtended','Extended').setExtended( 3.6e-4951);
5219: 'ESBmaxExtended','Extended').setExtended( 1.1e+4932);
5220: 'ESBminCurrency','Currency').SetExtended( - 922337203685477.5807);
5221: 'ESBmaxCurrency','Currency').SetExtended( 922337203685477.5807);
5222: 'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5223: 'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5224: 'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5225: 'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);

```



```

5226: 'ESBSqrtPi', 'Extended').setExtended( 1.77245385090551602729);
5227: 'ESBCbrt2', 'Extended').setExtended( 1.2599210498948731648);
5228: 'ESBCbrt3', 'Extended').setExtended( 1.4422495703074083823);
5229: 'ESBCbrt10', 'Extended').setExtended( 2.1544346900318837219);
5230: 'ESBCbrt100', 'Extended').setExtended( 4.6415888336127788924);
5231: 'ESBCbrtPi', 'Extended').setExtended( 1.4645918875615232630);
5232: 'ESBInvSqrt2', 'Extended').setExtended( 0.70710678118654752440);
5233: 'ESBInvSqrt3', 'Extended').setExtended( 0.57735026918962576451);
5234: 'ESBInvSqrt5', 'Extended').setExtended( 0.44721359549995793928);
5235: 'ESBInvSqrtPi', 'Extended').setExtended( 0.56418958354775628695);
5236: 'ESBInvCbrtPi', 'Extended').setExtended( 0.68278406325529568147);
5237: 'ESBe', 'Extended').setExtended( 2.7182818284590452354);
5238: 'ESBe2', 'Extended').setExtended( 7.3890560989306502272);
5239: 'ESBePi', 'Extended').setExtended( 23.140692632779269006);
5240: 'ESBePiOn2', 'Extended').setExtended( 4.8104773809653516555);
5241: 'ESBePiOn4', 'Extended').setExtended( 2.1932800507380154566);
5242: 'ESBLn2', 'Extended').setExtended( 0.69314718055994530942);
5243: 'ESBLn10', 'Extended').setExtended( 2.30258509299404568402);
5244: 'ESBLnPi', 'Extended').setExtended( 1.14472988584940017414);
5245: 'ESBLog10Base2', 'Extended').setExtended( 3.3219280948873623478);
5246: 'ESBLog2Base10', 'Extended').setExtended( 0.30102999566398119521);
5247: 'ESBLog3Base10', 'Extended').setExtended( 0.47712125471966243730);
5248: 'ESBLogPiBase10', 'Extended').setExtended( 0.4971498726941339);
5249: 'ESBLogEBase10', 'Extended').setExtended( 0.43429448190325182765);
5250: 'ESBPi', 'Extended').setExtended( 3.1415926535897932385);
5251: 'ESBInvPi', 'Extended').setExtended( 3.1830988618379067154e-1);
5252: 'ESBTwoPi', 'Extended').setExtended( 6.2831853071795864769);
5253: 'ESBThreePi', 'Extended').setExtended( 9.4247779607693797153);
5254: 'ESBPi2', 'Extended').setExtended( 9.8696044010893586188);
5255: 'ESBPiToE', 'Extended').setExtended( 22.459157718361045473);
5256: 'ESBPiOn2', 'Extended').setExtended( 1.5707963267948966192);
5257: 'ESBPiOn3', 'Extended').setExtended( 1.0471975511965977462);
5258: 'ESBPiOn4', 'Extended').setExtended( 0.7853981633974483096);
5259: 'ESBThreePiOn2', 'Extended').setExtended( 4.7123889803846898577);
5260: 'ESBFourPiOn3', 'Extended').setExtended( 4.1887902047863909846);
5261: 'ESBTwoToPower63', 'Extended').setExtended( 9223372036854775808.0);
5262: 'ESBOneRadian', 'Extended').setExtended( 57.295779513082320877);
5263: 'ESBOneDegree', 'Extended').setExtended( 1.7453292519943295769E-2);
5264: 'ESBOneMinute', 'Extended').setExtended( 2.9088820866572159615E-4);
5265: 'ESBOneSecond', 'Extended').setExtended( 4.8481368110953599359E-6);
5266: 'ESBGamma', 'Extended').setExtended( 0.57721566490153286061);
5267: 'ESBLnRt2Pi', 'Extended').setExtended( 9.189385332046727E-1);
5268: //LongWord', 'Cardinal
5269: TBitList', 'Word
5270: Function UMul( const Num1, Num2 : LongWord) : LongWord
5271: Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5272: Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5273: Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5274: Function SameFloat( const X1, X2 : Extended) : Boolean
5275: Function FloatIsZero( const X : Extended) : Boolean
5276: Function FloatIsPositive( const X : Extended) : Boolean
5277: Function FloatIsNegative( const X : Extended) : Boolean
5278: Procedure IncLim( var B : Byte; const Limit : Byte)
5279: Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5280: Procedure IncLimW( var B : Word; const Limit : Word)
5281: Procedure IncLimI( var B : Integer; const Limit : Integer)
5282: Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5283: Procedure DecLim( var B : Byte; const Limit : Byte)
5284: Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5285: Procedure DecLimW( var B : Word; const Limit : Word)
5286: Procedure DecLimI( var B : Integer; const Limit : Integer)
5287: Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5288: Function MaxB( const B1, B2 : Byte) : Byte
5289: Function MinB( const B1, B2 : Byte) : Byte
5290: Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5291: Function MinSI( const B1, B2 : ShortInt) : ShortInt
5292: Function MaxW( const B1, B2 : Word) : Word
5293: Function MinW( const B1, B2 : Word) : Word
5294: Function esbMaxI( const B1, B2 : Integer) : Integer
5295: Function esbMinI( const B1, B2 : Integer) : Integer
5296: Function MaxL( const B1, B2 : LongInt) : LongInt
5297: Function MinL( const B1, B2 : LongInt) : LongInt
5298: Procedure SwapB( var B1, B2 : Byte)
5299: Procedure SwapSI( var B1, B2 : ShortInt)
5300: Procedure SwapW( var B1, B2 : Word)
5301: Procedure SwapI( var B1, B2 : SmallInt)
5302: Procedure SwapL( var B1, B2 : LongInt)
5303: Procedure SwapI32( var B1, B2 : Integer)
5304: Procedure SwapC( var B1, B2 : LongWord)
5305: Procedure SwapInt64( var X, Y : Int64)
5306: Function esbSign( const B : LongInt) : ShortInt
5307: Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5308: Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5309: Function Max3Word( const X1, X2, X3 : Word) : Word
5310: Function Min3Word( const X1, X2, X3 : Word) : Word
5311: Function MaxBArray( const B : array of Byte) : Byte
5312: Function MaxWArray( const B : array of Word) : Word
5313: Function MaxSIArray( const B : array of ShortInt) : ShortInt
5314: Function MaxIArray( const B : array of Integer) : Integer

```

```

5315: Function MaxLArray( const B : array of LongInt ) : LongInt
5316: Function MinBArray( const B : array of Byte ) : Byte
5317: Function MinWArray( const B : array of Word ) : Word
5318: Function MinSArray( const B : array of ShortInt ) : ShortInt
5319: Function MinIArray( const B : array of Integer ) : Integer
5320: Function MinLArray( const B : array of LongInt ) : LongInt
5321: Function SumBArray( const B : array of Byte ) : Byte
5322: Function SumBArray2( const B : array of Byte ) : Word
5323: Function SumSArray( const B : array of ShortInt ) : ShortInt
5324: Function SumSArray2( const B : array of ShortInt ) : Integer
5325: Function SumWArray( const B : array of Word ) : Word
5326: Function SumWArray2( const B : array of Word ) : LongInt
5327: Function SumIArray( const B : array of Integer ) : Integer
5328: Function SumLArray( const B : array of LongInt ) : LongInt
5329: Function SumLWArray( const B : array of LongWord ) : LongWord
5330: Function ESBDigits( const X : LongWord ) : Byte
5331: Function BitsHighest( const X : LongWord ) : Integer
5332: Function ESBBitsNeeded( const X : LongWord ) : Integer
5333: Function esbGCD( const X, Y : LongWord ) : LongWord
5334: Function esbLCM( const X, Y : LongInt ) : Int64
5335: //Function esbLCM( const X, Y : LongInt ) : LongInt
5336: Function RelativePrime( const X, Y : LongWord ) : Boolean
5337: Function Get87ControlWord : TBitList
5338: Procedure Set87ControlWord( const CWord : TBitList )
5339: Procedure SwapExt( var X, Y : Extended )
5340: Procedure SwapDbl( var X, Y : Double )
5341: Procedure SwapSing( var X, Y : Single )
5342: Function esbSgn( const X : Extended ) : ShortInt
5343: Function Distance( const X1, Y1, X2, Y2 : Extended ) : Extended
5344: Function ExtMod( const X, Y : Extended ) : Extended
5345: Function ExtRem( const X, Y : Extended ) : Extended
5346: Function CompMOD( const X, Y : Comp ) : Comp
5347: Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended )
5348: Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended )
5349: Function DMS2Extended( const Degs, Mins, Secs : Extended ) : Extended
5350: Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended )
5351: Function MaxExt( const X, Y : Extended ) : Extended
5352: Function MinExt( const X, Y : Extended ) : Extended
5353: Function MaxEArray( const B : array of Extended ) : Extended
5354: Function MinEArray( const B : array of Extended ) : Extended
5355: Function MaxSArray( const B : array of Single ) : Single
5356: Function MinSArray( const B : array of Single ) : Single
5357: Function MaxCArray( const B : array of Comp ) : Comp
5358: Function MinCArray( const B : array of Comp ) : Comp
5359: Function SumSArray( const B : array of Single ) : Single
5360: Function SumEArray( const B : array of Extended ) : Extended
5361: Function SumSqEArray( const B : array of Extended ) : Extended
5362: Function SumSqDiffEArray( const B : array of Extended; Diff : Extended ) : Extended
5363: Function SumXYEArray( const X, Y : array of Extended ) : Extended
5364: Function SumCArray( const B : array of Comp ) : Comp
5365: Function FactorialX( A : LongWord ) : Extended
5366: Function PermutationX( N, R : LongWord ) : Extended
5367: Function esbBinomialCoeff( N, R : LongWord ) : Extended
5368: Function IsPositiveEArray( const X : array of Extended ) : Boolean
5369: Function esbGeometricMean( const X : array of Extended ) : Extended
5370: Function esbHarmonicMean( const X : array of Extended ) : Extended
5371: Function ESBMean( const X : array of Extended ) : Extended
5372: Function esbSampleVariance( const X : array of Extended ) : Extended
5373: Function esbPopulationVariance( const X : array of Extended ) : Extended
5374: Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended )
5375: Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended )
5376: Function GetMedian( const SortedX : array of Extended ) : Extended
5377: Function GetMode( const SortedX : array of Extended; var Mode : Extended ) : Boolean
5378: Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended )
5379: Function ESBMagnitude( const X : Extended ) : Integer
5380: Function ESBTan( Angle : Extended ) : Extended
5381: Function ESBcot( Angle : Extended ) : Extended
5382: Function ESBcosec( const Angle : Extended ) : Extended
5383: Function ESBsec( const Angle : Extended ) : Extended
5384: Function ESBArcTan( X, Y : Extended ) : Extended
5385: Procedure ESBsineCos( Angle : Extended; var SinX, CosX : Extended )
5386: Function ESBArcCos( const X : Extended ) : Extended
5387: Function ESBArcSin( const X : Extended ) : Extended
5388: Function ESBArcSec( const X : Extended ) : Extended
5389: Function ESBArcCosec( const X : Extended ) : Extended
5390: Function ESBLog10( const X : Extended ) : Extended
5391: Function ESBLog2( const X : Extended ) : Extended
5392: Function ESBLogBase( const X, Base : Extended ) : Extended
5393: Function Pow2( const X : Extended ) : Extended
5394: Function IntPow( const Base : Extended; const Exponent : LongWord ) : Extended
5395: Function ESBIntPower( const X : Extended; const N : LongInt ) : Extended
5396: Function XtoY( const X, Y : Extended ) : Extended
5397: Function esbTenToY( const Y : Extended ) : Extended
5398: Function esbTwoToY( const Y : Extended ) : Extended
5399: Function LogXtoBaseY( const X, Y : Extended ) : Extended
5400: Function esbISqrt( const I : LongWord ) : LongWord
5401: Function ILog2( const I : LongWord ) : LongWord
5402: Function IGreatestPowerOf2( const N : LongWord ) : LongWord
5403: Function ESBArcCosh( X : Extended ) : Extended

```

```

5404: Function ESBarSinh( X : Extended) : Extended
5405: Function ESBarTanh( X : Extended) : Extended
5406: Function ESBCosh( X : Extended) : Extended
5407: Function ESBSinh( X : Extended) : Extended
5408: Function ESBTanh( X : Extended) : Extended
5409: Function InverseGamma( const X : Extended) : Extended
5410: Function esbGamma( const X : Extended) : Extended
5411: Function esbLnGamma( const X : Extended) : Extended
5412: Function esbBeta( const X, Y : Extended) : Extended
5413: Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5414: end;
5415:
5416: *****Integer Huge Cardinal Utils*****
5417: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5418: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5419: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5420: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5421: Function BitCount_8( Value : byte) : integer
5422: Function BitCount_16( Value : uint16) : integer
5423: Function BitCount_32( Value : uint32) : integer
5424: Function BitCount_64( Value : uint64) : integer
5425: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5426: Procedure ( CountPrimalityTests : integer)
5427: Function gcd( a, b : THugeCardinal) : THugeCardinal
5428: Function lcm( a, b : THugeCardinal) : THugeCardinal
5429: Function isCoPrime( a, b : THugeCardinal) : boolean
5430: Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5431: Function hasSmallFactor( p : THugeCardinal) : boolean
5432: //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
NumbersTested: integer) : boolean
5433: Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5434: Const('StandardExponent','LongInt'( 65537));
5435: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
Numbers
5436: Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5437:
5438: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5439: begin
5440:   AddTypeS('TXRTLInteger', 'array of Integer
5441:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5442:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5443:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5444:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5445:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5446:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5447:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5448:   'BitsPerByte','LongInt'( 8);
5449:   BitsPerDigit','LongInt'( 32);
5450:   SignBitMask','LongWord( $80000000);
5451: Function XRTLAdjustBits( const ABits : Integer) : Integer
5452: Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5453: Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5454: Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5455: Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5456: Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5457: Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5458: Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5459: Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5460: Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5461: Function XRTLSignExtend(const AInteger:TXRTLInteger;ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5462: Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5463: Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5464: Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5465: Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5466: Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5467: Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5468: Procedure XRTLZero( var AInteger : TXRTLInteger)
5469: Procedure XRTLOne( var AInteger : TXRTLInteger)
5470: Procedure XRTLMOne( var AInteger : TXRTLInteger)
5471: Procedure XRTLTwo( var AInteger : TXRTLInteger)
5472: Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5473: Function XRTLABs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5474: Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5475: Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5476: Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5477: Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5478: Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5479: Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5480: Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5481: Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5482: Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5483: Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5484: Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5485: Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5486: Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5487: Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5488: Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHighApproxResult:TXRTLInteger)

```

```

5489: Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5490: Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5491: Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5492: Procedure XRTLSLBB(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5493: Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5494: Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5495: Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5496: Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5497: Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5498: Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5499: Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5500: Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5501: Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5502: Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5503: Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5504: Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5505: Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5506: Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5507: Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5508: Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5509: Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5510: Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5511: Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5512: Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5513: Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5514: Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5515: Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5516: Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5517: Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5518: Procedure XRTLMin1( const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5519: Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5520: Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5521: Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5522: Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5523: Procedure XRTLFactorial( const AInteger: TXRTLInteger; var AResult : TXRTLInteger)
5524: Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5525: end;
5526:
5527: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5528: begin
5529:   Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5530:   Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5531:   Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5532:   Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5533:   Procedure JvXPDrawBoundLines(const ACanvas:TCanvas;const BoundLines:TJvXPBoundLines;const
      AColor:TColor;const Rect:TRect);
5534:   Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5535:   Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5536:   Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect;const TopColor,BottomColor:TColor;const
      Swapped:Boolean);
5537:   Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5538:   Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5539:   Procedure JvXPPlaceText( const AParent: TControl; const ACanvas : TCanvas; const AText : TCaption; const
      AFont : TFont; const AEnabled,AShowAccelChar:Boolean;const AAlignment:TAlignment;const
      AWordWrap:Boolean;var Rect:TRect)
5540: end;
5541:
5542:
5543: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5544: begin
5545:   Function StrDec( S : String) : String
5546:   Function uIsNumeric( var S : String; var X : Float) : Boolean
5547:   Function ReadNumFromEdit( Edit : TEdit) : Float
5548:   Procedure WriteNumToFile( var F : Text; X : Float)
5549: end;
5550:
5551: procedure SIRegister_utexplore(CL: TPSPascalCompiler);
5552: begin
5553:   Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5554:   Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5555:   Procedure TeX_LeaveGraphics( Footer : Boolean)
5556:   Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5557:   Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5558:   Procedure TeX_SetGraphTitle( Title : String)
5559:   Procedure TeX_SetOxTitle( Title : String)
5560:   Procedure TeX_SetOyTitle( Title : String)
5561:   Procedure TeX_PlotOxAxis
5562:   Procedure TeX_PlotOyAxis
5563:   Procedure TeX_PlotGrid( Grid : TGrid)
5564:   Procedure TeX_WriteGraphTitle
5565:   Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5566:   Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5567:   Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5568:   Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5569:   Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)

```



```

5570: Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5571: Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5572: Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5573: Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5574: Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5575: Function Xcm( X : Float ) : Float
5576: Function Ycm( Y : Float ) : Float
5577: end;
5578:
5579: *-----*)
5580: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5581: begin
5582:   TConstArray', 'array of TVarRec
5583:   Function CopyVarRec( const Item : TVarRec ) : TVarRec
5584:   Function CreateConstArray( const Elements : array of const ) : TConstArray
5585:   Procedure FinalizeVarRec( var Item : TVarRec)
5586:   Procedure FinalizeConstArray( var Arr : TConstArray)
5587: end;
5588:
5589: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5590: begin
5591:   Function HexBS( B : Byte ) : ShortString
5592:   Function HexWS( W : Word ) : ShortString
5593:   Function HexLS( L : LongInt ) : ShortString
5594:   Function HexPtrS( P : Pointer ) : ShortString
5595:   Function BinaryBS( B : Byte ) : ShortString
5596:   Function BinaryWS( W : Word ) : ShortString
5597:   Function BinaryLS( L : LongInt ) : ShortString
5598:   Function OctalBS( B : Byte ) : ShortString
5599:   Function OctalWS( W : Word ) : ShortString
5600:   Function OctalLS( L : LongInt ) : ShortString
5601:   Function Str2Int16S( const S : ShortString; var I : SmallInt ) : Boolean
5602:   Function Str2WordS( const S : ShortString; var I : Word ) : Boolean
5603:   Function Str2LongS( const S : ShortString; var I : LongInt ) : Boolean
5604:   Function Str2RealS( const S : ShortString; var R : Double ) : Boolean
5605:   Function Str2Reals( const S : ShortString; var R : Real ) : Boolean
5606:   Function Str2ExtS( const S : ShortString; var R : Extended ) : Boolean
5607:   Function Long2StrS( L : LongInt ) : ShortString
5608:   Function Real2StrS( R : Double; Width : Byte; Places : ShortInt ) : ShortString
5609:   Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt ) : ShortString
5610:   Function ValPrepS( const S : ShortString ) : ShortString
5611:   Function CharStrS( C : AnsiChar; Len : Cardinal ) : ShortString
5612:   Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5613:   Function PadS( const S : ShortString; Len : Cardinal ) : ShortString
5614:   Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5615:   Function LeftPadS( const S : ShortString; Len : Cardinal ) : ShortString
5616:   Function TrimLeadS( const S : ShortString ) : ShortString
5617:   Function TrimTrails( const S : ShortString ) : ShortString
5618:   Function TrimS( const S : ShortString ) : ShortString
5619:   Function TrimSpacesS( const S : ShortString ) : ShortString
5620:   Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5621:   Function CenterS( const S : ShortString; Len : Cardinal ) : ShortString
5622:   Function EntabS( const S : ShortString; TabSize : Byte ) : ShortString
5623:   Function DetabS( const S : ShortString; TabSize : Byte ) : ShortString
5624:   Function ScrambleS( const S, Key : ShortString ) : ShortString
5625:   Function SubstituteS( const S, FromStr, ToStr : ShortString ) : ShortString
5626:   Function FilterS( const S, Filters : ShortString ) : ShortString
5627:   Function CharExistsS( const S : ShortString; C : AnsiChar ) : Boolean
5628:   Function CharCountS( const S : ShortString; C : AnsiChar ) : Byte
5629:   Function WordCountS( const S, WordDelims : ShortString ) : Cardinal
5630:   Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal ) : Boolean
5631:   Function ExtractWords( N : Cardinal; const S, WordDelims : ShortString ) : ShortString
5632:   Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar ) : Cardinal
5633:   Function AsciiPositionS( N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5634:   Function ExtractAsciiS( N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5635:   Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
Margin:Cardinal;PadToMargin:Boolean)
5636:   Function CompStringS( const S1, S2 : ShortString ) : Integer
5637:   Function CompUCStringS( const S1, S2 : ShortString ) : Integer
5638:   Function SoundexS( const S : ShortString ) : ShortString
5639:   Function MakeLetterSetS( const S : ShortString ) : Longint
5640:   Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5641:   Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;
5642:   Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;
5643:   Function DefaultExtensionS( const Name, Ext : ShortString ) : ShortString
5644:   Function ForceExtensionS( const Name, Ext : ShortString ) : ShortString
5645:   Function JustFileNameS( const PathName : ShortString ) : ShortString
5646:   Function JustNameS( const PathName : ShortString ) : ShortString
5647:   Function JustExtensionS( const Name : ShortString ) : ShortString
5648:   Function JustPathNameS( const PathName : ShortString ) : ShortString
5649:   Function AddBackSlashS( const DirName : ShortString ) : ShortString
5650:   Function CleanPathNameS( const PathName : ShortString ) : ShortString
5651:   Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal ) : Boolean
5652:   Function CommaizeS( L : LongInt ) : ShortString
5653:   Function CommaizeChS( L : Longint; Ch : AnsiChar ) : ShortString
5654:   Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:ShortString;Sep,
DecPt:Char):ShortString;

```

```

5655: Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
RtCurr:ShortString;Sep:AnsiChar):ShortString;
5656: Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5657: Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5658: Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5659: Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5660: Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5661: Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5662: Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5663: Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5664: Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5665: Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5666: Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5667: Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5668: Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5669: Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
SubString:ShortString):Bool;
5670: Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
SubStr:ShortString):Bool;
5671: Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5672: Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5673: Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5674: Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5675: Function ExtractTokensS(const S,
Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5676: Function IsChAlphaS( C : Char) : Boolean
5677: Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5678: Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5679: Function IsStrAlphaS( const S : ShortString) : Boolean
5680: Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5681: Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5682: Function LastWords( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5683: Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5684: Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5685: Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5686: Function KeepCharsS( const S, Chars : ShortString) : ShortString
5687: Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
Cardinal):ShortString;
5688: Function ReplaceStringS(const S,OldString,NewString:ShortString;N:Cardinal;var
Replacements:Cardinal):ShortString;
5689: Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5690: Function ReplaceWords(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
Replacements:Cardinal):ShortString
5691: Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
Replacements:Cardinal):ShortString
5692: Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5693: Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5694: Function TrimCharsS( const S, Chars : ShortString) : ShortString
5695: Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5696: end;
5697:
5698:
5699: *****unit uPSI_StUtils; from Systools4*****
5700: Function SignL( L : LongInt) : Integer
5701: Function SignF( F : Extended) : Integer
5702: Function MinWord( A, B : Word) : Word
5703: Function MidWord( W1, W2, W3 : Word) : Word
5704: Function MaxWord( A, B : Word) : Word
5705: Function MinLong( A, B : LongInt) : LongInt
5706: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5707: Function MaxLong( A, B : LongInt) : LongInt
5708: Function MinFloat( F1, F2 : Extended) : Extended
5709: Function MidFloat( F1, F2, F3 : Extended) : Extended
5710: Function MaxFloat( F1, F2 : Extended) : Extended
5711: Function MakeInteger16( H, L : Byte) : SmallInt
5712: Function MakeWordS( H, L : Byte) : Word
5713: Function SwapNibble( B : Byte) : Byte
5714: Function SwapWord( L : LongInt) : LongInt
5715: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5716: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5717: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5718: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5719: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5720: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5721: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5722: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5723: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5724: Procedure ExchangeBytes( var I, J : Byte)
5725: Procedure ExchangeWords( var I, J : Word)
5726: Procedure ExchangeLongInts( var I, J : LongInt)
5727: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5728: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5729: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5730: Function AddWordToPtr( P : __Pointer; W : Word) : __Pointer
5731: //*****uPSI_StFIN;*****
5732: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5733: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended

```

```

5734: Function BondDuration( Settlement,Maturity:TStDate;Rate,
Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5735: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,
Redemption:Ext;Freq:TStFrequency;Basis:TStBasis): Extended
5736: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5737: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5738: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis ) : LongInt
5739: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer ) : Extended
5740: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5741: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer ) : Extended
5742: Function DollarToDecimalText( DecDollar : Extended ) : string
5743: Function DollarToFraction( DecDollar : Extended; Fraction : Integer ) : Extended
5744: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer ) : string
5745: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency ) : Extended
5746: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
PV:Extended;Freq:TStFrequency;Timing:TStPaymentTime):Extended;
5747: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double ) : Extended
5748: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer ) : Extended
5749: Function InterestRates(NPeriods:Int;Pmt,PV,
FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5750: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended ) : Extended
5751: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended ) : Extended
5752: Function IsCardValid( const S : string ) : Boolean
5753: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5754: Function ModifiedIRR(const Values : array of Double; FinanceRate,ReinvestRate: Extended ) : Extended
5755: Function ModifiedIRR16(const Values, NValues:Integer; FinanceRate,ReinvestRate: Extended ) : Extended
5756: Function NetPresentValueS( Rate : Extended; const Values : array of Double ) : Extended
5757: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer ) : Extended
5758: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency ) : Extended
5759: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5760: Function NonperiodicNPV(Rate:Extended;const Values:array of Double;const Dates:array of TStDate):Extended;
5761: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
: TStPaymentTime ) : Extended
5762: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5763: Function PresentValues( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
Timing: TStPaymentTime): Extended
5764: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5765: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean ) : Extended
5766: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5767: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5768: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended ) : Extended
5769: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
Factor : Extended; NoSwitch : boolean ) : Extended
5770: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5771: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5772: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5773:
5774: *****unit uPSI_StAstroP;
5775: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5776: *****unit unit uPSI_StStat; Statistic Package of SysTools*****
5777: Function AveDev( const Data : array of Double ) : Double
5778: Function AveDev16( const Data, NData : Integer ) : Double
5779: Function Confidence( Alpha, StandardDev : Double; Size : LongInt ) : Double
5780: Function Correlation( const Data1, Data2 : array of Double ) : Double
5781: Function Correlation16( const Data1, Data2, NData : Integer ) : Double
5782: Function Covariance( const Data1, Data2 : array of Double ) : Double
5783: Function Covariance16( const Data1, Data2, NData : Integer ) : Double
5784: Function DevSq( const Data : array of Double ) : Double
5785: Function DevSql6( const Data, NData : Integer ) : Double
5786: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5787: ///Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5788: Function GeometricMeanS( const Data : array of Double ) : Double
5789: Function GeometricMean16( const Data, NData : Integer ) : Double
5790: Function HarmonicMeanS( const Data : array of Double ) : Double
5791: Function HarmonicMean16( const Data, NData : Integer ) : Double
5792: Function Largest( const Data : array of Double; K : Integer ) : Double
5793: Function Largest16( const Data, NData : Integer; K : Integer ) : Double
5794: Function MedianS( const Data : array of Double ) : Double
5795: Function Median16( const Data, NData : Integer ) : Double
5796: Function Mode( const Data : array of Double ) : Double
5797: Function Model16( const Data, NData : Integer ) : Double
5798: Function Percentile( const Data : array of Double; K : Double ) : Double
5799: Function Percentile16( const Data, NData : Integer; K : Double ) : Double
5800: Function PercentRank( const Data : array of Double; X : Double ) : Double
5801: Function PercentRank16( const Data, NData : Integer; X : Double ) : Double
5802: Function Permutations( Number, NumberChosen : Integer ) : Extended
5803: Function Combinations( Number, NumberChosen : Integer ) : Extended
5804: Function Factorials( N : Integer ) : Extended
5805: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean ) : Integer
5806: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean ) : Integer
5807: Function Smallest( const Data : array of Double; K : Integer ) : Double
5808: Function Smallest16( const Data, NData : Integer; K : Integer ) : Double
5809: Function TrimMean( const Data : array of Double; Percent : Double ) : Double
5810: Function TrimMean16( const Data, NData : Integer; Percent : Double ) : Double
5811: AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'

```

```

5812:   +1 : Double; R2 : Double; sigma :Double; SSR: double; SSE: Double; F0 : Double; df : Integer;end
5813: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
LF:TstLinEst;ErrorStats:Bool;
5814: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
LF:TstLinEst;ErrorStats:Bool;
5815: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5816: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5817: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5818: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5819: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5820: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5821: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5822: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5823: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5824: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5825: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5826: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5827: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5828: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5829: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5830: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5831: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5832: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5833: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5834: Function NormSDist( Z : Single) : Single
5835: Function NormSInv( Probability : Single) : Single
5836: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5837: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5838: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5839: Function Erfc( X : Single) : Single
5840: Function GammaLn( X : Single) : Single
5841: Function LargestSort( const Data : array of Double; K : Integer) : Double
5842: Function SmallestSort( const Data : array of double; K : Integer) : Double
5843:
5844: procedure SIRegister_TstSorter(CL: TPSPascalCompiler);
5845:   Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5846:   Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5847:   Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5848:   Function DefaultMergeName( MergeNum : Integer) : string
5849:   Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5850:
5851: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5852: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5853: Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5854: Function SunPos( UT : TStDateTimeRec) : TStPosRec
5855: Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5856: Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5857: Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TstTwilight):TStRiseSetRec
5858: Function LunarPhase( UT : TStDateTimeRec) : Double
5859: Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5860: Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5861: Function FirstQuarter( D : TStDate) : TStLunarRecord
5862: Function FullMoon( D : TStDate) : TStLunarRecord
5863: Function LastQuarter( D : TStDate) : TStLunarRecord
5864: Function NewMoon( D : TStDate) : TStLunarRecord
5865: Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5866: Function NextFullMoon( D : TStDate) : TStDateTimeRec
5867: Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5868: Function NextNewMoon( D : TStDate) : TStDateTimeRec
5869: Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5870: Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5871: Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5872: Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5873: Function SiderealTime( UT : TStDateTimeRec) : Double
5874: Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5875: Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5876: Function SEaster( Y, Epoch : Integer) : TStDate
5877: Function DateTimeToAJD( D : TDateTime) : Double
5878: Function HoursMin( RA : Double) : ShortString
5879: Function DegsMin( DC : Double) : ShortString
5880: Function AJDToDateTime( D : Double) : TDateTime
5881:
5882: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5883: Function CurrentDate : TStDate
5884: Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5885: Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5886: Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5887: Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5888: Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5889: Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5890: Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5891: Function WeekOfYear( Julian : TStDate) : Byte
5892: Function AstJulianDate( Julian : TStDate) : Double
5893: Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5894: Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5895: Function StDayOfWeek( Julian : TStDate) : TStDayType
5896: Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5897: Function StIsLeapYear( Year : Integer) : Boolean
5898: Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer

```



```

5899: Function ResolveEpoch( Year, Epoch : Integer) : Integer
5900: Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5901: Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5902: Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5903: Function CurrentTime : TStTime
5904: Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5905: Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5906: Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5907: Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5908: Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5909: Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5910: Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5911: Function DateTimeToStDate( DT : TDateTime) : TStDate
5912: Function DateTimeToStTime( DT : TDateTime) : TStTime
5913: Function StDateToDateTime( D : TStDate) : TDateTime
5914: Function StTimeToDateTime( T : TStTime) : TDateTime
5915: Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5916: Function Convert4ByteDate( FourByteDate : TStDate) : Word
5917:
5918: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5919: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5920: Function MonthToString( const Month : Integer) : string
5921: Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5922: Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5923: Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5924: Function DayOfWeekToString( const WeekDay : TStDayType) : string
5925: Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5926: Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5927: Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5928: Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5929: Function TimeStringToStTime( const Picture, S : string) : TStTime
5930: Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5931: Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5932: Function DateStringIsBlank( const Picture, S : string) : Boolean
5933: Function InternationalDate( ForceCentury : Boolean) : string
5934: Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
5935: Function InternationalTime( ShowSeconds : Boolean) : string
5936: Procedure ResetInternationalInfo
5937:
5938: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5939: Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
5940: Function AnsiUpperCaseShort32( const S : string) : string
5941: Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
5942: Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
5943: Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
5944: Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5945: Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
5946: Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
5947: Function Upcase( C : AnsiChar) : AnsiChar
5948: Function LoCase( C : AnsiChar) : AnsiChar
5949: Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
5950: Function CompStruct( const S1, S2, Size : Cardinal) : Integer
5951: Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
5952: Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5953: Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5954: Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
5955: Procedure RaiseContainerError( Code : longint)
5956: Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
5957: Function ProductOverflow( A, B : LongInt) : Boolean
5958: Function StNewStr( S : string) : PShortString
5959: Procedure StDisposeStr( PS : PShortString)
5960: Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
5961: Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
5962: Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
5963: Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
5964: Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
5965: Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
5966:
5967: procedure SIRegister_usvd(CL: TPSPascalCompiler);
5968: begin
5969: Procedure SV_DeComp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
5970: Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
5971: Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
5972: Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
5973: Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
5974: end;
5975:
5976: *****unit unit ; StMath Package of SysTools*****
5977: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
5978: Function PowerS( Base, Exponent : Extended) : Extended
5979: Function StInvCos( X : Double) : Double
5980: Function StInvSin( Y : Double) : Double
5981: Function StInvTan2( X, Y : Double) : Double
5982: Function StTan( A : Double) : Double
5983: Procedure DumpException; unit StExpEng;
5984: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
5985:
5986: *****unit unit ; StCRC Package of SysTools*****
5987: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt

```

```

5988: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
5989: Function Adler32OfFile( FileName : AnsiString ) : LongInt
5990: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
5991: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
5992: Function Crc16OfFile( FileName : AnsiString ) : Cardinal
5993: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
5994: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
5995: Function Crc32OfFile( FileName : AnsiString ) : LongInt
5996: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
5997: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
5998: Function InternetSumOfFile( FileName : AnsiString ) : Cardinal
5999: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
6000: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
6001: Function Kermit16OfFile( FileName : AnsiString ) : Cardinal
6002:
6003: //*****unit unit ; StBCD Package of SysTools*****
6004: Function AddBcd( const B1, B2 : TbcdS ) : TbcdS
6005: Function SubBcd( const B1, B2 : TbcdS ) : TbcdS
6006: Function MulBcd( const B1, B2 : TbcdS ) : TbcdS
6007: Function DivBcd( const B1, B2 : TbcdS ) : TbcdS
6008: Function ModBcd( const B1, B2 : TbcdS ) : TbcdS
6009: Function NegBcd( const B : TbcdS ) : TbcdS
6010: Function AbsBcd( const B : TbcdS ) : TbcdS
6011: Function FracBcd( const B : TbcdS ) : TbcdS
6012: Function IntBcd( const B : TbcdS ) : TbcdS
6013: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal ) : TbcdS
6014: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal ) : TbcdS
6015: Function ValBcd( const S : string ) : TbcdS
6016: Function LongBcd( L : LongInt ) : TbcdS
6017: Function ExtBcd( E : Extended ) : TbcdS
6018: Function ExpBcd( const B : TbcdS ) : TbcdS
6019: Function LnBcd( const B : TbcdS ) : TbcdS
6020: Function IntPowBcd( const B : TbcdS; E : LongInt ) : TbcdS
6021: Function PowBcd( const B, E : TbcdS ) : TbcdS
6022: Function SqrtBcd( const B : TbcdS ) : TbcdS
6023: Function CmpBcd( const B1, B2 : TbcdS ) : Integer
6024: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal ) : Boolean
6025: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal ) : Boolean
6026: Function IsIntBcd( const B : TbcdS ) : Boolean
6027: Function TruncBcd( const B : TbcdS ) : LongInt
6028: Function BcdExt( const B : TbcdS ) : Extended
6029: Function RoundBcd( const B : TbcdS ) : LongInt
6030: Function StrBcd( const B : TbcdS; Width, Places : Cardinal ) : string
6031: Function StrExpBcd( const B : TbcdS; Width : Cardinal ) : string
6032: Function FormatBcd( const Format : string; const B : TbcdS ) : string
6033: Function StrGeneralBcd( const B : TbcdS ) : string
6034: Function FloatFormBcd( const Mask: string; B: TbcdS; const LtCurr, RtCurr: string; Sep, DecPt: AnsiChar ) : string
6035: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6036:
6037: //*****unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools*****
6038: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6039: Function StFieldTypeToStr( FieldType : TStSchemaFieldType ) : AnsiString
6040: Function StStrToFieldType( const S : AnsiString ) : TStSchemaFieldType
6041: Function StDeEscape( const EscStr : AnsiString ) : Char
6042: Function StDoEscape( Delim : Char ) : AnsiString
6043: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char ) : AnsiString
6044: Function AnsiHashText( const S : string; Size : Integer ) : Integer
6045: Function AnsiHashStr( const S : string; Size : Integer ) : Integer
6046: Function AnsiELFHashText( const S : string; Size : Integer ) : Integer
6047: Function AnsiELFHashStr( const S : string; Size : Integer ) : Integer
6048:
6049: //*****unit unit ; StNetCon Package of SysTools*****
6050: with AddClassN( FindClass( 'TStComponent' ), 'TStNetConnection' ) do begin
6051:   Constructor Create( AOwner : TComponent )
6052:   Function Connect : DWord
6053:   Function Disconnect : DWord
6054:   RegisterProperty( 'Password', 'String', iptrw );
6055:   Property( 'UserName', 'String', iptrw );
6056:   Property( 'ConnectOptions', 'TStNetConnectOptionsSet', iptrw );
6057:   Property( 'DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw );
6058:   Property( 'LocalDevice', 'String', iptrw );
6059:   Property( 'ServerName', 'String', iptrw );
6060:   Property( 'ShareName', 'String', iptrw );
6061:   Property( 'OnConnect', 'TNotifyEvent', iptrw );
6062:   Property( 'OnConnectFail', 'TOnConnectFailEvent', iptrw );
6063:   Property( 'OnConnectCancel', 'TOnConnectCancelEvent', iptrw );
6064:   Property( 'OnDisconnect', 'TNotifyEvent', iptrw );
6065:   Property( 'OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw );
6066:   Property( 'OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw );
6067: end;
6068: //*****Thread Functions Context of Win API --- more objects in SyncObjs.pas
6069: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6070: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection )
6071: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection )
6072: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection )
6073: Function InitializeCriticalSectionAndSpinCount( var
  lpCriticalSection: TRTLCriticalSection; dwSpinCount: DWORD ): BOOL;
6074: Function SetCriticalSectionSpinCount( var lpCriticalSection: TRTLCriticalSection; dwSpinCount: DWORD ): DWORD;
6075: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection ) : BOOL

```

```

6076: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6077: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6078: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6079: Function SuspendThread( hThread : THandle) : DWORD
6080: Function ResumeThread( hThread : THandle) : DWORD
6081: Function CreateThread2(ThreadFunc: TThreadFunction2; thrId: DWord) : THandle
6082: Function GetCurrentThread : THandle
6083: Procedure ExitThread( dwExitCode : DWORD)
6084: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6085: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6086: Procedure EndThread(ExitCode: Integer);
6087: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6088: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6089: Procedure FreeProcInstance( Proc : FARPROC)
6090: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6091: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6092: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6093: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6094: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6095: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6096: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6097: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6098: Function CurrentParallelJobInfo : TParallelJobInfo
6099: Function ObtainParallelJobInfo : TParallelJobInfo
6100:
6101: *****unit uPSI_JclMime;
6102: Function MimeEncodeString( const S : AnsiString) : AnsiString
6103: Function MimeDecodeString( const S : AnsiString) : AnsiString
6104: Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6105: Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6106: Function MimeEncodedSize( const I : Cardinal) : Cardinal
6107: Function MimeDecodedSize( const I : Cardinal) : Cardinal
6108: Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6109: Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6110: Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6111: Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
Cardinal;
6112:
6113: *****unit uPSI_JclPrint;
6114: Procedure DirectPrint( const Printer, Data : string)
6115: Procedure SetPrinterPixelsPerInch
6116: Function GetPrinterResolution : TPoint
6117: Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6118: Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6119:
6120:
6121: //*****unit uPSI_ShLwApi;*****
6122: Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6123: Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6124: Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6125: Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6126: Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6127: Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6128: Function StrDup( lpSrch : PChar) : PChar
6129: Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6130: Function StrFormatKBSIZE( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6131: Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6132: Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6133: Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6134: Function StrPBrk( psz, pszSet : PChar) : PChar
6135: Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6136: Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6137: Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6138: Function StrSpn( psz, pszSet : PChar) : Integer
6139: Function StrStr( lpFirst, lpSrch : PChar) : PChar
6140: Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6141: Function StrToInt( lpSrch : PChar) : Integer
6142: Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6143: Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6144: Function ChrCmpI( w1, w2 : WORD) : BOOL
6145: Function ChrCmpIA( w1, w2 : WORD) : BOOL
6146: Function ChrCmpIW( w1, w2 : WORD) : BOOL
6147: Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6148: Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6149: Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6150: Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6151: Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6152: Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6153: SZ_CONTENTTYPE_HTMLA, 'String 'text/html
6154: SZ_CONTENTTYPE_HTMLW, 'String 'text/html
6155: SZ_CONTENTTYPE_HTML, 'string SZ_CONTENTTYPE_HTMLA);
6156: SZ_CONTENTTYPE_CDF, 'String 'application/x-cdf
6157: SZ_CONTENTTYPE_CDFW, 'String 'application/x-cdf
6158: SZ_CONTENTTYPE_CDF, 'string SZ_CONTENTTYPE_CDF);
6159: Function PathIsHTMLFile( pszPath : PChar) : BOOL
6160: STIF_DEFAULT, 'LongWord( $00000000);
6161: STIF_SUPPORT_HEX, 'LongWord( $00000001);
6162: Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer

```

```

6163: Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer ) : PChar
6164: Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer ) : PChar
6165: Function PathAddBackslash( pszPath : PChar ) : PChar
6166: Function PathAddExtension( pszPath : PChar; pszExt : PChar ) : BOOL
6167: Function PathAppend( pszPath : PChar; pMore : PChar ) : BOOL
6168: Function PathBuildRoot( szRoot : PChar; iDrive : Integer ) : PChar
6169: Function PathCanonicalize( pszBuf : PChar; pszPath : PChar ) : BOOL
6170: Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar ) : PChar
6171: Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT ) : BOOL
6172: Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD ) : BOOL
6173: Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar ) : Integer
6174: Function PathFileExists( pszPath : PChar ) : BOOL
6175: Function PathFindExtension( pszPath : PChar ) : PChar
6176: Function PathFindFileName( pszPath : PChar ) : PChar
6177: Function PathFindNextComponent( pszPath : PChar ) : PChar
6178: Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar ) : BOOL
6179: Function PathGetArgs( pszPath : PChar ) : PChar
6180: Function PathFindSuffixArray( pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6181: Function PathIsLFNFileSpec( lpName : PChar ) : BOOL
6182: Function PathGetCharType( ch : Char ) : UINT
6183: GCT_INVALID, 'LongWord( $0000);
6184: GCT_LFNCHAR, 'LongWord( $0001);
6185: GCT_SHORTCHAR, 'LongWord( $0002);
6186: GCT_WILD, 'LongWord( $0004);
6187: GCT_SEPARATOR, 'LongWord( $0008);
6188: Function PathGetDriveNumber( pszPath : PChar ) : Integer
6189: Function PathIsDirectory( pszPath : PChar ) : BOOL
6190: Function PathIsDirectoryEmpty( pszPath : PChar ) : BOOL
6191: Function PathIsFileSpec( pszPath : PChar ) : BOOL
6192: Function PathIsPrefix( pszPrefix, pszPath : PChar ) : BOOL
6193: Function PathIsRelative( pszPath : PChar ) : BOOL
6194: Function PathIsRoot( pszPath : PChar ) : BOOL
6195: Function PathIsSameRoot( pszPath1, pszPath2 : PChar ) : BOOL
6196: Function PathIsUNC( pszPath : PChar ) : BOOL
6197: Function PathIsNetworkPath( pszPath : PChar ) : BOOL
6198: Function PathIsUNCServer( pszPath : PChar ) : BOOL
6199: Function PathIsUNCServerShare( pszPath : PChar ) : BOOL
6200: Function PathIsContentType( pszPath, pszContentType : PChar ) : BOOL
6201: Function PathIsURL( pszPath : PChar ) : BOOL
6202: Function PathMakePretty( pszPath : PChar ) : BOOL
6203: Function PathMatchSpec( pszFile, pszSpec : PChar ) : BOOL
6204: Function PathParseIconLocation( pszIconFile : PChar ) : Integer
6205: Procedure PathQuoteSpaces( lpsz : PChar)
6206: Function PathRelativePathTo( pszPath:PChar; pszFrom:PChar; dwAttrFrom:DWORD; pszTo:PChar; dwAttrTo:DWORD ):BOOL;
6207: Procedure PathRemoveArgs( pszPath : PChar)
6208: Function PathRemoveBackslash( pszPath : PChar ) : PChar
6209: Procedure PathRemoveBlanks( pszPath : PChar)
6210: Procedure PathRemoveExtension( pszPath : PChar)
6211: Function PathRemoveFileSpec( pszPath : PChar ) : BOOL
6212: Function PathRenameExtension( pszPath : PChar; pszExt : PChar ) : BOOL
6213: Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6214: Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6215: Function PathSkipRoot( pszPath : PChar ) : PChar
6216: Procedure PathStripPath( pszPath : PChar)
6217: Function PathStripToRoot( pszPath : PChar ) : BOOL
6218: Procedure PathUnquoteSpaces( lpsz : PChar)
6219: Function PathMakeSystemFolder( pszPath : PChar ) : BOOL
6220: Function PathUnmakeSystemFolder( pszPath : PChar ) : BOOL
6221: Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD ) : BOOL
6222: Procedure PathUndecorate( pszPath : PChar)
6223: Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6224: URL_SCHEME_INVALID, 'LongInt'( - 1);
6225: URL_SCHEME_UNKNOWN, 'LongInt'( 0);
6226: URL_SCHEME_FTP, 'LongInt'( 1);
6227: URL_SCHEME_HTTP, 'LongInt'( 2);
6228: URL_SCHEME_GOPHER, 'LongInt'( 3);
6229: URL_SCHEME_MAILTO, 'LongInt'( 4);
6230: URL_SCHEME_NEWS, 'LongInt'( 5);
6231: URL_SCHEME_NNTP, 'LongInt'( 6);
6232: URL_SCHEME_TELNET, 'LongInt'( 7);
6233: URL_SCHEME_WAIS, 'LongInt'( 8);
6234: URL_SCHEME_FILE, 'LongInt'( 9);
6235: URL_SCHEME_MK, 'LongInt'( 10);
6236: URL_SCHEME_HTTPS, 'LongInt'( 11);
6237: URL_SCHEME_SHELL, 'LongInt'( 12);
6238: URL_SCHEME_SNEWS, 'LongInt'( 13);
6239: URL_SCHEME_LOCAL, 'LongInt'( 14);
6240: URL_SCHEME_JAVASCRIPT, 'LongInt'( 15);
6241: URL_SCHEME_VBSCRIPT, 'LongInt'( 16);
6242: URL_SCHEME_ABOUT, 'LongInt'( 17);
6243: URL_SCHEME_RES, 'LongInt'( 18);
6244: URL_SCHEME_MAXVALUE, 'LongInt'( 19);
6245: URL_SCHEME, 'Integer
6246: URL_PART_NONE, 'LongInt'( 0);
6247: URL_PART_SCHEME, 'LongInt'( 1);
6248: URL_PART_HOSTNAME, 'LongInt'( 2);
6249: URL_PART_USERNAME, 'LongInt'( 3);
6250: URL_PART_PASSWORD, 'LongInt'( 4);
6251: URL_PART_PORT, 'LongInt'( 5);

```



```

6252: URL_PART_QUERY', 'LongInt'( 6);
6253: URL_PART', 'DWORD
6254: URLIS_URL', 'LongInt'( 0);
6255: URLIS_OPAQUE', 'LongInt'( 1);
6256: URLIS_NOHISTORY', 'LongInt'( 2);
6257: URLIS_FILEURL', 'LongInt'( 3);
6258: URLIS_APPLIABLE', 'LongInt'( 4);
6259: URLIS_DIRECTORY', 'LongInt'( 5);
6260: URLIS_HASQUERY', 'LongInt'( 6);
6261: TUrlIs', 'DWORD
6262: URL_UNESCAPE', 'LongWord( $10000000);
6263: URL_ESCAPE_UNSAFE', 'LongWord( $20000000);
6264: URL_PLUGGABLE_PROTOCOL', 'LongWord( $40000000);
6265: URL_WININET_COMPATIBILITY', 'LongWord( DWORD ( $80000000 ));
6266: URL_DONT_ESCAPE_EXTRA_INFO', 'LongWord( $02000000);
6267: URL_ESCAPE_SPACES_ONLY', 'LongWord( $04000000);
6268: URL_DONT_SIMPLIFY', 'LongWord( $08000000);
6269: URL_NO_META', 'longword( URL_DONT_SIMPLIFY);
6270: URL_UNESCAPE_INPLACE', 'LongWord( $00100000);
6271: URL_CONVERT_IF_DOSPATH', 'LongWord( $00200000);
6272: URL_UNESCAPE_HIGH_ANSI_ONLY', 'LongWord( $00400000);
6273: URL_INTERNAL_PATH', 'LongWord( $00800000);
6274: URL_FILE_USE_PATHURL', 'LongWord( $00010000);
6275: URL_ESCAPE_PERCENT', 'LongWord( $00001000);
6276: URL_ESCAPE_SEGMENT_ONLY', 'LongWord( $00002000);
6277: URL_PARTFLAG_KEEPScheme', 'LongWord( $00000001);
6278: URL_APPLY_DEFAULT', 'LongWord( $00000001);
6279: URL_APPLY_GUESSScheme', 'LongWord( $00000002);
6280: URL_APPLY_GUESSFILE', 'LongWord( $00000004);
6281: URL_APPLY_FORCEAPPLY', 'LongWord( $00000008);
6282: Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL ) : Integer
6283: Function UrlCombine(pszBase, pszRelative: PChar; pszCombin: PChar; out pcchCombin: DWORD; dwFlags: DWORD): HRESULT;
6284: Function UrlCanonicalize(pszUrl: PChar; pszCanonicalized: PChar; pcchCanonic: DWORD; dwFlags: DWORD): HRESULT;
6285: Function UrlIsOpaque( pszURL : PChar ) : BOOL
6286: Function UrlIsNoHistory( pszURL : PChar ) : BOOL
6287: Function UrlIsFileUrl( pszURL : PChar ) : BOOL
6288: Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs ) : BOOL
6289: Function UrlGetLocation( psz1 : PChar ) : PChar
6290: Function UrlUnescape( pszUrl, pszUnescaped : PChar; pcchUnescaped: DWORD; dwFlags : DWORD ) : HRESULT
6291: Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped: DWORD; dwFlags : DWORD) : HRESULT
6292: Function UrlCreateFromPath(pszPath: PChar; pszUrl: PChar; pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6293: Function PathCreateFromUrl(pszUrl: PChar; pszPath: PChar; pcchPath: DWORD; dwFlags : DWORD) : HRESULT
6294: Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6295: Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart, dwFlags: DWORD) : HRESULT
6296: Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6297: Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6298: Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6299: Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6300: Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6301: Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6302: Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar ) : DWORD
6303: Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6304: Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName: PChar; var
pcchValueName: DWORD; pdwType: DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6305: Function SHQueryInfoKey(hKey: HKEY; pcSubKeys, pcchMaxSubKeyLen, pcVal, pcchMaxValueNameLen: DWORD): Longint;
6306: Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6307: Function SHRegGetPath(hKey: HKEY; pcszSubKey, pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6308: Function SHRegSetPath( hKey: HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6309: SHREGDEL_DEFAULT', 'LongWord( $00000000);
6310: SHREGDEL_HKCU', 'LongWord( $00000001);
6311: SHREGDEL_HKLM', 'LongWord( $00000010);
6312: SHREGDEL_BOTH', 'LongWord( $00000011);
6313: SHREGENUM_DEFAULT', 'LongWord( $00000000);
6314: SHREGENUM_HKCU', 'LongWord( $00000001);
6315: SHREGENUM_HKLM', 'LongWord( $00000010);
6316: SHREGENUM_BOTH', 'LongWord( $00000011);
6317: SHREGSET_HKCU', 'LongWord( $00000001);
6318: SHREGSET_FORCE_HKCU', 'LongWord( $00000002);
6319: SHREGSET_HKLM', 'LongWord( $00000004);
6320: SHREGSET_FORCE_HKLM', 'LongWord( $00000008);
6321: TSHRegDelFlags', 'DWORD
6322: TSHRegEnumFlags', 'DWORD
6323: HUSKEY', 'THandle
6324: ASSOCF_INIT_NOREMAPCLSID', 'LongWord( $00000001);
6325: ASSOCF_INIT_BYEXENAME', 'LongWord( $00000002);
6326: ASSOCF_OPEN_BYEXENAME', 'LongWord( $00000002);
6327: ASSOCF_INIT_DEFAULTTOSTAR', 'LongWord( $00000004);
6328: ASSOCF_INIT_DEFAULTTOFOLDER', 'LongWord( $00000008);
6329: ASSOCF_NOUSERSETTINGS', 'LongWord( $00000010);
6330: ASSOCF_NOTRUNCATE', 'LongWord( $00000020);
6331: ASSOCF_VERIFY', 'LongWord( $00000040);
6332: ASSOCF_REMAPRUNDLL', 'LongWord( $00000080);
6333: ASSOCF_NOFIXUPS', 'LongWord( $00000100);
6334: ASSOCF_IGNOREBASECLASS', 'LongWord( $00000200);
6335: ASSOCF', 'DWORD
6336: ASSOCTSTR_COMMAND', 'LongInt'( 1);
6337: ASSOCTSTR_EXECUTABLE', 'LongInt'( 2);
6338: ASSOCTSTR_FRIENDLYDOCNAME', 'LongInt'( 3);
6339: ASSOCTSTR_FRIENDLYAPPNAME', 'LongInt'( 4);

```

```

6340: ASSOCSTR_NOOPEN', 'LongInt'( 5);
6341: ASSOCSTR_SHELLNEWVALUE', 'LongInt'( 6);
6342: ASSOCSTR_DDECOMMAND', 'LongInt'( 7);
6343: ASSOCSTR_DDEIFEXEC', 'LongInt'( 8);
6344: ASSOCSTR_DDEAPPLICATION', 'LongInt'( 9);
6345: ASSOCSTR_DDETOPIC', 'LongInt'( 10);
6346: ASSOCSTR_INFOTIP', 'LongInt'( 11);
6347: ASSOCSTR_MAX', 'LongInt'( 12);
6348: ASSOCSTR', 'DWORD
6349: ASSOCKEY_SHELLEXECCLASS', 'LongInt'( 1);
6350: ASSOCKEY_APP', 'LongInt'( 2);
6351: ASSOCKEY_CLASS', 'LongInt'( 3);
6352: ASSOCKEY_BASECLASS', 'LongInt'( 4);
6353: ASSOCKEY_MAX', 'LongInt'( 5);
6354: ASSOCKEY', 'DWORD
6355: ASSOCDATA_MSIDESCRIPTOR', 'LongInt'( 1);
6356: ASSOCDATA_NOACTIVATEHANDLER', 'LongInt'( 2);
6357: ASSOCDATA_QUERYCLASSSTORE', 'LongInt'( 3);
6358: ASSOCDATA_HASPERUSERASSOC', 'LongInt'( 4);
6359: ASSOCDATA_MAX', 'LongInt'( 5);
6360: ASSOCDATA', 'DWORD
6361: ASSOCENUM_NONE', 'LongInt'( 0);
6362: ASSOCENUM', 'DWORD
6363: SID_IQueryAssociations', 'String' '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6364: SHACF_DEFAULT $00000000);
6365: SHACF_FILESYSTEM', 'LongWord( $00000001);
6366: SHACF_URLHISTORY', 'LongWord( $00000002);
6367: SHACF_URLMRU', 'LongWord( $00000004);
6368: SHACF_USETAB', 'LongWord( $00000008);
6369: SHACF_FILESYS_ONLY', 'LongWord( $00000010);
6370: SHACF_AUTOSUGGEST_FORCE_ON', 'LongWord( $10000000);
6371: SHACF_AUTOSUGGEST_FORCE_OFF', 'LongWord( $20000000);
6372: SHACF_AUTOAPPEND_FORCE_ON', 'LongWord( $40000000);
6373: SHACF_AUTOAPPEND_FORCE_OFF', 'LongWord( DWORD ( $80000000 ));
6374: Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD ) : HRESULT
6375: Procedure SHSetThreadRef( punk : IUnknown)
6376: Procedure SHGetThreadRef( out ppunk : IUnknown)
6377: CTF_INSIST', 'LongWord( $00000001);
6378: CTF_THREAD_REF', 'LongWord( $00000002);
6379: CTF_PROCESS_REF', 'LongWord( $00000004);
6380: CTF_COINIT', 'LongWord( $00000008);
6381: Function SHCreateShellPalette( hdc : HDC ) : HPALETTE
6382: Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6383: Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6384: Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6385: Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6386: Function DrawFocusRect( hdc : HDC; const lprc : TRect) : BOOL
6387: Function FillRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6388: Function FrameRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6389: Function InvertRect( hdc : HDC; const lprc : TRect) : BOOL
6390: Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6391: Function SetRectEmpty( var lprc : TRect) : BOOL
6392: Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6393: Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6394: Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6395: Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6396:
6397: Function InitializeFlatSB( hWnd : HWND) : Bool
6398: Procedure UninitializeFlatSB( hWnd : HWND)
6399: Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6400: Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6401: Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word //of JvWin32
6402: Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6403: Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6404: Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6405: Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6406:
6407:
6408: // ***** 204 unit uPSI_ShellAPI;
6409: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6410: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6411: Procedure DragFinish( Drop : HDROP)
6412: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6413: Function ShellExecute( hWnd : HWND; Operation, FileName, Parameters, Directory : PChar; ShowCmd : Integer) : HINST
6414: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6415: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6416: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6417: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6418: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6419: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6420: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6421: Function ExtractIconEx( lpszFile : PChar; nIconIndex : Int; var phiconLarge, phiconSmall : HICON; nIcons : UINT) : UINT;
6422: Procedure SHFreeNameMappings( hNameMappings : THandle)
6423:
6424: DLLVER_PLATFORM_WINDOWS', 'LongWord( $00000001);
6425: DLLVER_PLATFORM_NT', 'LongWord( $00000002);
6426: DLLVER_MAJOR_MASK', 'LongWord( Int64 ( $FFFF000000000000 ));
6427: DLLVER_MINOR_MASK', 'LongWord( Int64 ( $0000FFFF00000000 ));
6428: DLLVER_BUILD_MASK', 'LongWord( Int64 ( $00000000FFFF0000 ));

```

```

6429: DLLVER_QFE_MASK', 'LongWord( Int64 ( $000000000000FFFF ));
6430: Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6431: Function SimpleXMLEncode( const S : string) : string
6432: Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6433: Function XMLEncode( const S : string) : string
6434: Function XMLDecode( const S : string) : string
6435: Function EntityEncode( const S : string) : string
6436: Function EntityDecode( const S : string) : string
6437:
6438: procedure RIRegister_CPort_Routines(S: TPSExec);
6439: Procedure EnumComPorts( Ports : TStrings)
6440: Procedure ListComPorts( Ports : TStrings)
6441: Procedure ComPorts( Ports : TStrings) //Alias to Arduino
6442: Function GetComPorts: TStringlist;
6443: Function StrToBaudRate( Str : string) : TBaudRate
6444: Function StrToStopBits( Str : string) : TStopBits
6445: Function StrToDataBits( Str : string) : TDataBits
6446: Function StrToParity( Str : string) : TParityBits
6447: Function StrToFlowControl( Str : string) : TFlowControl
6448: Function BaudRateToStr( BaudRate : TBaudRate) : string
6449: Function StopBitsToStr( StopBits : TStopBits) : string
6450: Function DataBitsToStr( DataBits : TDataBits) : string
6451: Function ParityToStr( Parity : TParityBits) : string
6452: Function FlowControlToStr( FlowControl : TFlowControl) : string
6453: Function ComErrorsToStr( Errors : TComErrors) : String
6454:
6455: Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wParamFilterMin, wParamFilterMax : UINT) : BOOL
6456: Function DispatchMessage( const lpMsg : TMsg) : Longint
6457: Function TranslateMessage( const lpMsg : TMsg) : BOOL
6458: Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6459: Function PeekMessage( var lpMsg: TMsg; hWnd: HWND; wParamFilterMin, wParamFilterMax, wRemoveMsg: UINT): BOOL
6460: Function GetMessagePos : DWORD
6461: Function GetMessageTime : Longint
6462: Function GetMessageExtraInfo : Longint
6463: Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6464: Procedure JAddToRecentDocs( const Filename : string)
6465: Procedure ClearRecentDocs
6466: Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6467: Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6468: Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6469: Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6470: Function RecycleFile( FileToRecycle : string) : Boolean
6471: Function JCopyFile( FromFile, ToDir : string) : Boolean
6472: Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6473: Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6474: Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
  DWORD; var pcbBytesNeeded : DWORD) : BOOL
6475: Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
  DWORD; var pcbBytesNeeded : DWORD) : BOOL
6476: Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
  DWORD; var pcbBytesNeeded : DWORD) : BOOL
6477: Function EnumServicesStatusExA( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
  dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned,
  lpResumeHandle : DWORD; pszGroupName : LP
6478: Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
  dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned,
  lpResumeHandle : DWORD; pszGroupNam
6479: Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
  dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned,
  lpResumeHandle : DWORD; pszGroupName
6480: Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6481:
6482: ***** unit uPSI_JclPeImage;
6483:
6484: Function IsValidPeFile( const FileName : TFileName) : Boolean
6485: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6486: Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6487: Function PeRebaseImage( const ImageName : TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
  DWORD) : TJclRebaseImageInfo
6488: Function PeVerifyChecksum( const FileName : TFileName) : Boolean
6489: Function PeClearChecksum( const FileName : TFileName) : Boolean
6490: Function PeUpdateChecksum( const FileName : TFileName) : Boolean
6491: Function PeDoesExportFunction( const FileName : TFileName; const
  FuncName : string; Options : TJclSmartCompOptions) : Bool;
6492: Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
  ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6493: Function PeIsExportFunctionForwarded( const FileName : TFileName; const
  FunctionName : string; Options : TJclSmartCompOptions) : Bool
6494: Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
  : string; Options : TJclSmartCompOptions) : Boolean
6495: Function PeDoesImportLibrary( const FileName : TFileName; const
  LibraryName : string; Recursive : Boolean) : Boolean;
6496: Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
  Boolean; FullPathName : Boolean) : Boolean
6497: Function PeImportedFunctions( const FileName : TFileName; const FunctionsList : TStrings; const
  LibraryName : string; IncludeLibNames : Boolean) : Boolean
6498: Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6499: Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6500: Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean

```

```

6501: Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
NamesList:TStrings):Bool
6502: Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6503: Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
Descript:Bool):Bool;
6504: Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6505: Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6506: Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6507: //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6508: //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6509: //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6510: //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
PImageSectionHeader
6511: //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6512: //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6513: Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
____Pointer;
6514: SIRegister_TJclPeSectionStream(CL);
6515: SIRegister_TJclPeMapImgHookItem(CL);
6516: SIRegister_TJclPeMapImgHooks(CL);
6517: //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
NtHeaders:TImageNtHeaders):Boolean
6518: //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6519: Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6520: TJclBorUmSymbolModifier','( smQualified, smLinkProc )
6521: TJclBorUmSymbolModifiers',' 'set of TJclBorUmSymbolModifier
6522: TJclBorUmDescription',' 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6523: TJclBorUmResult',' ( urOk, urNotMangled, urMicrosoft, urError )
6524: TJclPeUmResult',' ( umNotMangled, umBorland, umMicrosoft )
6525: Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6526: Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
Descript:TJclBorUmDescription):TJclBorUmResult;
6527: Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6528: Function PeBorUnmangleName3( const Name : string) : string;
6529: Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6530: Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6531:
6532:
6533: //***** SysTools uPSI_StSystem; *****
6534: Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6535: Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6536: Function DeleteVolumeLabel( Drive : Char) : Cardinal
6537: //Procedure EnumerateDirectories(const
StartDir:AnsiString;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6538: //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
IncludeItem:TIncludeItemFunc);
6539: Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6540: Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6541: Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTImeRec
6542: Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6543: Function FlushOsBuffers( Handle : Integer) : Boolean
6544: Function GetCurrentUser : AnsiString
6545: Function GetDiskClass( Drive : Char) : DiskClass
6546: Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
SectorsPerCluster:Cardinal):Bool;
6547: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
DiskSize:Double):Bool;
6548: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
DiskSize:Comp):Boolean;
6549: { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes }
6550: Function getDiskSpace2(const path: String; index: integer): int64;
6551: Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6552: Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6553: Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6554: Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6555: Function GetLongPath( const APath : AnsiString) : AnsiString
6556: Function GetMachineName : AnsiString
6557: Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6558: Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6559: Function GetShortPath( const APath : AnsiString) : AnsiString
6560: Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6561: Function GetTempFolder( aForceSlash : boolean) : AnsiString
6562: Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6563: Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6564: Function GlobalDateTImeToLocal( const UTC : TStDateTImeRec; MinOffset : Integer) : TStDateTImeRec
6565: Function StIsDirectory( const DirName : AnsiString) : Boolean
6566: Function IsDirectoryEmpty( const S : AnsiString) : Integer
6567: Function IsDriveReady( Drive : Char) : Boolean
6568: Function IsFile( const FileName : AnsiString) : Boolean
6569: Function IsFileArchive( const S : AnsiString) : Integer
6570: Function IsFileHidden( const S : AnsiString) : Integer
6571: Function IsFileReadOnly( const S : AnsiString) : Integer
6572: Function IsFileSystem( const S : AnsiString) : Integer
6573: Function LocalDateTImeToGlobal( const DT1 : TStDateTImeRec; MinOffset : Integer) : TStDateTImeRec
6574: Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6575: Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6576: Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6577: Procedure SplitPath( const APath : AnsiString; Parts : TStrings)

```



```

6578: Function StDateTimeToFileTime( const FileTime : TStDateTimeRec ) : LongInt
6579: Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec ) : Longint
6580: Function UnixTimeToStDateTime( UnixTime : Longint ) : TStDateTimeRec
6581: Function ValidDrive( Drive : Char ) : Boolean
6582: Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char ) : Cardinal
6583:
6584: //*****unit uPSI_JellANMan;*****
6585: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
const PasswordNeverExpires : Boolean ) : Boolean
6586: Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
const PasswordNeverExpires : Boolean ) : Boolean
6587: Function DeleteAccount( const Servername, Username : string ) : Boolean
6588: Function DeleteLocalAccount( Username : string ) : Boolean
6589: Function CreateLocalGroup( const Server, Groupname, Description : string ) : Boolean
6590: Function CreateGlobalGroup( const Server, Groupname, Description : string ) : Boolean
6591: Function DeleteLocalGroup( const Server, Groupname : string ) : Boolean
6592: Function GetLocalGroups( const Server : string; const Groups : TStrings ) : Boolean
6593: Function GetGlobalGroups( const Server : string; const Groups : TStrings ) : Boolean
6594: Function LocalGroupExists( const Group : string ) : Boolean
6595: Function GlobalGroupExists( const Server, Group : string ) : Boolean
6596: Function AddAccountToLocalGroup( const Accountname, Groupname : string ) : Boolean
6597: Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID ) : string
6598: Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string )
6599: Function IsLocalAccount( const AccountName : string ) : Boolean
6600: Function TimeStampInterval( StartStamp, EndStamp : TDateTime ) : integer
6601: Function GetRandomString( NumChar : cardinal ) : string
6602:
6603: //*****unit uPSI_CUtils;*****
6604: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6605: Function cIsWinNT : boolean
6606: Procedure cFilesFromWildcard(Directory,Mask : string;var Files:TStringList;Subdirs,ShowDirs,
Multitasking:Boolean;
6607: Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ) : THandle
6608: Function cRunAndGetOutput( Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean ) : string
6609: Function cGetShortName( FileName : string ) : string
6610: Procedure cShowError( Msg : String )
6611: Function cCommaStrToStr( s : string; formatstr : string ) : string
6612: Function cIncludeQuoteIfSpaces( s : string ) : string
6613: Function cIncludeQuoteIfNeeded( s : string ) : string
6614: Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream )
6615: Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6616: Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET ) : boolean;
6617: Function cBuildFilter1( var value : string; const _filters : array of string ) : boolean;
6618: Function cCodeInstoStr( s : string ) : string
6619: Function cStrtoCodeIns( s : string ) : string
6620: Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6621: Function cAttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6622: Procedure cStrToPoint( var pt : TPoint; value : string )
6623: Function cPointtoStr( const pt : TPoint ) : string
6624: Function cListtoStr( const List : TStrings ) : string
6625: Function ListtoStr( const List : TStrings ) : string
6626: Procedure StrtoList( s : string; const List : TStrings; const delimiter : char )
6627: Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char )
6628: Function cGetFileType( const FileName : string ) : TUnitType
6629: Function cGetExTyp( const FileName : string ) : TExUnitType
6630: Procedure cSetPath( Add : string; const UseOriginal : boolean )
6631: Function cExpandFileto( const FileName : string; const BasePath : string ) : string
6632: Function cFileSamePath( const FileName : string; const TestPath : string ) : boolean
6633: Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem )
6634: Function cGetLastPos( const SubStr : string; const S : string ) : integer
6635: Function cGenMakePath( FileName : String ) : String;
6636: Function cGenMakePath2( FileName : String ) : String
6637: Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean ) : String;
6638: Function cGetRealPath( BrokenFileName : String; Directory : String ) : String
6639: Function cCalcMod( Count : Integer ) : Integer
6640: Function cGetVersionString( FileName : string ) : string
6641: Function cCheckChangeDir( var Dir : string ) : boolean
6642: Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6643: Function cIsNumeric( s : string ) : boolean
6644: Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6645: Function AttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6646: Function cGetFileType( const FileName : string ) : TUnitType
6647: Function Atoi(const aStr: string): integer
6648: Function Itoa(const aint: integer): string
6649:
6650:
6651: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6652: begin
6653:   FindClass('TOBJECT'),'EHTTP
6654:   FindClass('TOBJECT'),'EHTTPParser
6655:   //AnsiCharSet', 'set of AnsiChar
6656:   AnsiStringArray', 'array of AnsiString
6657:   THHTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6658:   THHTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6659:   THHTTPVersion', 'record Version : THHTTPVersionEnum; Protocol : TH
6660:   + 'THHTTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6661:   + 'CustomMinVersion : Integer; end
6662:   THHTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'

```

```

6663: + 'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6664: + 'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6665: + 'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6666: + 'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6667: + 'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6668: + 'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6669: + ' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6670: + 'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6671: + 'nection, hntOrigin, hntKeepAlive )
6672: THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6673: THTTPCustomHeader', 'record FieldName: AnsiString; FieldValue : '
6674: + ' AnsiString; end
6675: //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6676: THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6677: THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount: Int64; end
6678: //PHTTPContentLength', '^THTTPContentLength // will not work
6679: THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6680: THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6681: + 'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6682: + 'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6683: + 'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6684: + 'ationCustom, hctAudioCustom, hctVideoCustom )
6685: THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6686: + 'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6687: + ' CustomStr : AnsiString; end
6688: THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6689: THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek : '
6690: + ' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6691: + 'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6692: + 'String; DateTime : TDateTime; Custom : AnsiString; end
6693: THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6694: THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnum'
6695: + 'm; Custom : AnsiString; end
6696: THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6697: THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6698: + ' Custom : AnsiString; end
6699: THTTPPageFieldEnum', '( hafNone, hafCustom, hafAge )
6700: THTTPPageField', 'record Value : THTTPPageFieldEnum; Age : Int64; Custom: AnsiString; end
6701: THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6702: THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6703: + ', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6704: THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6705: + ', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6706: + 'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6707: THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6708: THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6709: + 'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6710: THTTPContentEncoding', 'record Value: THTTPContentEncodingEnum; Custom: AnsiString; end;
6711: THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6712: THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6713: + 'FieldEnum; List : array of THTTPContentEncoding; end
6714: THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6715: THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6716: + ' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6717: THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6718: THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6719: + 'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6720: THTTPSetCookieFieldEnum', '( hscNone, hscDecoded, hscCustom )
6721: THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6722: THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6723: THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6724: + 'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6725: + 'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6726: + 'CustomFieldArray; Custom : AnsiString; end
6727: //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6728: THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6729: THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6730: THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6731: //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6732: THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6733: THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6734: + ' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6735: THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6736: + 'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6737: + ' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6738: + '; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6739: THTTPCustomHeaders', 'array of THTTPCustomHeader
6740: //THTTFFixedHeaders', 'array[THTTPHeaderNameEnum] of AnsiString
6741: THTTFFixedHeaders', 'array[0..42] of AnsiString
6742: THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6743: + 'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6744: THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6745: THTTPRequestStartLine', 'record Method: THTTPMethod; URI: AnsiString; Version: THTTPVersion; end
6746: THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6747: + ' FixedHeaders : THTTFFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6748: + 'kie : THTTPCookieField; IfModifiedSince: THTTPDateField; IfUnmodifiedSince: THTTPDateField; end
6749: //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6750: THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6751: + ' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end

```

```

6752:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6753:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6754:   + 'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6755:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6756:   + 'FixedHeaders : THTTFFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6757:   + 'okies : THTTPSetCookieFieldArray; Expires : THTTPOdateField; LastModified : '
6758:   + 'THTTPOdateField; Age : THTTPOageField; end
6759:   //PHTTPOResponseHeader', '^THTTPOResponseHeader // will not work
6760:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6761:   + 'er : THTTPOResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6762:   Function HTTPMessageHasContent( const H : THTTPCommonHeaders ) : Boolean
6763:   Procedure InitHTTPRequest( var A : THTTPRequest)
6764:   Procedure InitHTTPResponse( var A : THTTPResponse)
6765:   Procedure ClearHTTPVersion( var A : THTTPVersion)
6766:   Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6767:   Procedure ClearHTTPContentType( var A : THTTPContentType)
6768:   Procedure ClearHTTPDateField( var A : THTTPOdateField)
6769:   Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6770:   Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6771:   Procedure ClearHTTPOageField( var A : THTTPOageField)
6772:   Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6773:   Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6774:   Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6775:   Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6776:   Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6777:   //Procedure ClearHTTFFixedHeaders( var A : THTTFFixedHeaders)
6778:   Procedure ClearHTTPCustomHeaders( var A : THTTPOCustomHeaders)
6779:   Procedure ClearHTTPOCookieField( var A : THTTPOCookieField)
6780:   Procedure ClearHTTPMethod( var A : THTTPOMethod)
6781:   Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6782:   Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6783:   Procedure ClearHTTPRequest( var A : THTTPRequest)
6784:   Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6785:   Procedure ClearHTTPResponseHeader( var A : THTTPOResponseHeader)
6786:   Procedure ClearHTTPResponse( var A : THTTPResponse)
6787:   THTTPOStringOption', '( hsoNone )
6788:   THTTPOStringOptions', 'set of THTTPOStringOption
6789:   FindClass( 'TOBJECT', 'TAnsiStringBuilder
6790:
6791:   Procedure BuildStrHTTPVersion(const A:THTTPVersion; const B:TAnsiStringBuilder;const
P:THTTPOStringOptions);
6792:   Procedure BuildStrHTTPContentLengthValue(const A: THTTPContentLength; const B : TAnsiStringBuilder; const
P : THTTPOStringOptions)
6793:   Procedure BuildStrHTTPContentLength( const A : THTTPContentLength; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6794:   Procedure BuildStrHTTPContentTypeValue( const A : THTTPContentType; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6795:   Procedure BuildStrHTTPContentType(const A:THTTPContentType;const B:TAnsiStringBuilder; const
P:THTTPOStringOptions)
6796:   Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
B : TAnsiStringBuilder; const P : THTTPOStringOptions)
6797:   Procedure BuildStrHTTPOdateFieldValue( const A : THTTPOdateField; const B : TAnsiStringBuilder; const P :
THTTPOStringOptions)
6798:   Procedure BuildStrHTTPOdateField(const A:THTTPOdateField;const B:TAnsiStringBuilder;const
P:THTTPOStringOptions);
6799:   Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
TAnsiStringBuilder; const P : THTTPOStringOptions)
6800:   Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6801:   Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6802:   Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6803:   Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6804:   Procedure BuildStrHTTPOageField(const A:THTTPOageField;const B:TAnsiStringBuilder;const
P:THTTPOStringOptions);
6805:   Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6806:   Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
B:TAnsiStringBuilder;const P:THTTPOStringOptions)
6807:   Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6808:   Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6809:   Procedure BuildStrHTTFFixedHeaders(const A:THTTFFixedHeaders;const B:TAnsiStrBuilder;const
P:THTTPOStringOptions)
6810:   Procedure BuildStrHTTPCustomHeaders( const A : THTTPOCustomHeaders; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6811:   Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6812:   Procedure BuildStrHTTPOCookieFieldValue( const A : THTTPOCookieField; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6813:   Procedure BuildStrHTTPOCookieField(const A:THTTPOCookieField;const B:TAnsiStringBuilder;const
P:THTTPOStringOptions);
6814:   Procedure BuildStrHTTPMethod( const A : THTTPOMethod; const B : TAnsiStringBuilder; const P :
THTTPOStringOptions)
6815:   Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)

```

```

6816: Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6817: Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6818: Procedure BuildStrHTTPResponseCookieFieldArray(const A : THTTPSetCookieFieldArray; const B :
TAnsiStringBuilder; const P : THTTPStringOptions)
6819: Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
THTTPStrOptions);
6820: Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
P:THTTPStringOptions);
6821: Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
P:THTTPStringOptions);
6822: Function HTTPContentTypeValueToStr(const A : THTTPContentType) : AnsiString
6823: Function HTTPSetCookieFieldValueToStr(const A : THTTPSetCookieField) : AnsiString
6824: Function HTTPCookieFieldValueToStr(const A : THTTPCookieField) : AnsiString
6825: Function HTTPMethodToStr(const A : THTTPMethod) : AnsiString
6826: Function HTTPRequestToStr(const A : THTTPRequest) : AnsiString
6827: Function HTTPResponseToStr(const A : THTTPResponse) : AnsiString
6828: Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT
6829: +PHeaderNameEnum; const HeaderPtr : __Pointer) : Boolean
6830: SIRegister_THTTPParser(CL);
6831: FindClass('TOBJECT'),'THTTPContentDecoder
6832: THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6833: THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsize )
6834: THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,
6835: + crcsContentCRLF, crcsTrailer, crcsFinished )
6836: THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6837: SIRegister_THTTPContentDecoder(CL);
6838: THTTPContentReaderMechanism', '( hcrnEvent, hcrnString, hcrnStream, hcrnFile )
6839: FindClass('TOBJECT'),'THTTPContentReader
6840: THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6841: THTTPContentReaderLogEvent', 'Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
LogLevel: Int;
6842: SIRegister_THTTPContentReader(CL);
6843: THTTPContentWriterMechanism', '(hctmEvent, hctmString, hctmStream, hctmFile )
6844: FindClass('TOBJECT'),'THTTPContentWriter
6845: THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6846: SIRegister_THTTPContentWriter(CL);
6847: Procedure SelfTestcHTTUtils
6848: end;
6849:
6850: (*-----*)
6851: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6852: begin
6853: 'TLSLibraryVersion','String '1.00
6854: 'TLSError_None','LongInt'( 0);
6855: 'TLSError_InvalidBuffer','LongInt'( 1);
6856: 'TLSError_InvalidParameter','LongInt'( 2);
6857: 'TLSError_InvalidCertificate','LongInt'( 3);
6858: 'TLSError_InvalidState','LongInt'( 4);
6859: 'TLSError_DecodeError','LongInt'( 5);
6860: 'TLSError_BadProtocol','LongInt'( 6);
6861: Function TLSErrorMessage(const TLSError : Integer) : String
6862: SIRegister_ETLSError(CL);
6863: TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6864: PTLSPProtocolVersion', '^TTLSProtocolVersion // will not work
6865: Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6866: Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6867: Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6868: Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6869: Function IsTLSProtocolVersion(const A, B : TTLSProtocolVersion) : Boolean
6870: Function IsSSL3(const A : TTLSProtocolVersion) : Boolean
6871: Function IsSSL3(const A : TTLSProtocolVersion) : Boolean
6872: Function IsTLS10(const A : TTLSProtocolVersion) : Boolean
6873: Function IsTLS11(const A : TTLSProtocolVersion) : Boolean
6874: Function IsTLS12(const A : TTLSProtocolVersion) : Boolean
6875: Function IsTLS10OrLater(const A : TTLSProtocolVersion) : Boolean
6876: Function IsTLS11OrLater(const A : TTLSProtocolVersion) : Boolean
6877: Function IsTLS12OrLater(const A : TTLSProtocolVersion) : Boolean
6878: Function IsFutureTLSVersion(const A : TTLSProtocolVersion) : Boolean
6879: Function IsKnownTLSVersion(const A : TTLSProtocolVersion) : Boolean
6880: Function TLSProtocolVersionToStr(const A : TTLSProtocolVersion) : String
6881: Function TLSProtocolVersionName(const A : TTLSProtocolVersion) : String
6882: PTLSRandom', '^TTLSRandom // will not work
6883: Procedure InitTLSSRandom( var Random : TTLSRandom)
6884: Function TLSRandomToStr(const Random : TTLSRandom) : AnsiString
6885: 'TLSSessionIDMaxLen','LongInt'( 32);
6886: Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6887: Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6888: Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6889: TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm
6890: +; Signature : TTLSSignatureAlgorithm; end
6891: // PTLSSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +// will not work
6892: TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6893: TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE
6894: +DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6895: TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma
6896: +HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )

```



```

6897:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I
6898:   +integer; Supported : Boolean; end
6899:   PTLSTMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6900:   'TLS_MAC_MAXDIGESTSIZE', 'LongInt'( 64);
6901:   TTLSPRFAlgorithm', '( tlpshaSHA256 )
6902:   Function tlp_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6903:   Function tlp_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6904:   Function tlp_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6905:   Function tlp_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6906:   Function tlp10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6907:   Function tlp12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6908:   Function tlp12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6909:   Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AStrng;const
Size:Int):AStrng;
6910:   Function tlp10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
Size:Integer):AnsiString
6911:   Function tlp12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
6912:   Function tlp12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
6913:   Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
ClientRandom : AnsiString; const Size : Integer) : AnsiString
6914:   Function tlp10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6915:   Function tlp12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6916:   Function tlp12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6917:   Function TLSTMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
ServerRandom:AnsiString) : AnsiString
6918:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr
6919:   +ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6920:   +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6921:   Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6922:   Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
6923:   'TLS_PLAINTEXT_FRAGMENT_MAXSIZE', 'LongInt'( 16384 - 1);
6924:   'TLS_COMPRESSED_FRAGMENT_MAXSIZE', 'LongInt'( 16384 + 1024);
6925:   Procedure SelfTestcTLSUtils
6926:   end;
6927:
6928:   (*-----*)
6929:   procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6930:   begin
6931:     sPosData', 'record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
integer; disks : integer; mx : integer; my : integer; end
6932:     // pBoard', '^tBoard // will not work
6933:     Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6934:     Function rCheckMove( color : byte; cx, cy : integer) : integer
6935:     //Function rDoStep( data : pBoard) : word
6936:     Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6937:     end;
6938:
6939:   procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
6940:   begin
6941:     Function InEditMode( ADataSet : TDataSet) : Boolean
6942:     Function CheckDataSource( ADataSource : TDataSource) : Boolean;
6943:     Function CheckDataSourceCel(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6944:     Function GetFieldText( AField : TField) : String
6945:     end;
6946:
6947:   procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
6948:   begin
6949:     TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6950:     TMyPrintRange', '( prAll, prSelected )
6951:     TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
6952:     +ded, ssDateTime, ssTime, ssCustom )
6953:     TSortDirection', '( sdAscending, sdDescending )
6954:     TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
6955:     TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ
6956:     +er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
6957:     TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
6958:     TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
6959:     SIRegister_TSortOptions(CL);
6960:     SIRegister_TPrintOptions(CL);
6961:     TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
6962:     SIRegister_TSortedList(CL);
6963:     TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
6964:     TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
6965:     TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
6966:     TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
6967:     +Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
6968:     SIRegister_TFontSetting(CL);
6969:     SIRegister_TFontList(CL);
6970:     AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :
6971:     + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
6972:     TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6973:     TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6974:     TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
6975:     TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
6976:     TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)

```

```

6977: TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
6978: TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : integer)
6979: + 'r'; var SortStyle : TSortStyle)
6980: TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :
6981: + ' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
6982: SIRegister_TSortGrid(CL);
6983: Function ExtendedCompare( const Str1, Str2 : String) : Integer
6984: Function NormalCompare( const Str1, Str2 : String) : Integer
6985: Function DateTimeCompare( const Str1, Str2 : String) : Integer
6986: Function NumericCompare( const Str1, Str2 : String) : Integer
6987: Function TimeCompare( const Str1, Str2 : String) : Integer
6988: //Function Compare( Item1, Item2 : Pointer) : Integer
6989: end;
6990:
6991: ***** procedure Register_IB(CL: TPSPascalCompiler);
6992: Procedure IBAlloc( var P, OldSize, NewSize : Integer)
6993: Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
6994: Procedure IBDataBaseError
6995: Function StatusVector : PISC_STATUS
6996: Function StatusVectorArray : PStatusVector
6997: Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
6998: Function StatusVectorAsText : string
6999: Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7000: Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7001:
7002:
7003: //*****unit uPSI_BoldUtils;*****
7004: Function CharCount( c : char; const s : string) : integer
7005: Function BoldNamesEqual( const name1, name2 : string) : Boolean
7006: Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7007: Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7008: Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7009: Function BoldTrim( const S : string) : string
7010: Function BoldIsPrefix( const S, Prefix : string) : Boolean
7011: Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7012: Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7013: Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7014: Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7015: Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7016: //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7017: Function CapitalisedToSpaced( Capitalised : String) : String
7018: Function SpacedToCapitalised( Spaced : String) : String
7019: Function BooleanToString( BoolValue : Boolean) : String
7020: Function StringToBoolean( StrValue : String) : Boolean
7021: Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7022: Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7023: Function StringListToVarArray( List : TStringList) : variant
7024: Function IsLocalMachine( const MachineName : WideString) : Boolean
7025: Function GetComputerNameStr : string
7026: Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7027: Function BoldStrToDateFmt( const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7028: Function BoldDateToStrFmt( const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7029: Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7030: Function BoldParseFormattedDate(const value:String;const formats:array of string; var
Date:TDateTime):Boolean;
7031: Procedure EnsureTrailing( var Str : String; ch : char)
7032: Function BoldDirectoryExists( const Name : string) : Boolean
7033: Function BoldForceDirectories( Dir : string) : Boolean
7034: Function BoldRootRegistryKey : string
7035: Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7036: Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7037: Function LogicalAnd( A, B : Integer) : Boolean
7038: record TByHandleFileInformation dwFileAttributes : DWORD; '
7039: + 'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7040: + ' : TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7041: + 'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7042: Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7043: Function IsFirstInstance : Boolean
7044: Procedure ActivateFirst( AString : PChar)
7045: Procedure ActivateFirstCommandLine
7046: function MakeAckPkt(const BlockNumber: Word): string;
7047: procedure SendError(UDPBase:TiUDPBase;APeerIP:string;const APort:Integer;const ErrNumber:Word;ErrStr:string;
7048: procedure SendError(UDPClient: TiUDPClient; const ErrNumber: Word; ErrorMessage: string); overload;
7049: procedure SendError(UDPBase: TiUDPBase; APeerIP: string; const APort: Integer; E: Exception); overload;
7050: procedure SendError(UDPClient: TiUDPClient; E: Exception); overload;
7051: function IdStrToWord(const Value: String): Word;
7052: function IdWordToStr(const Value: Word): WordStr;
7053: Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7054: Function CPUFeatures : TCPUFeatures
7055:
7056: procedure SIRegister_xrttl_util_CPUUtils(CL: TPSPascalCompiler);
7057: begin
7058: AddTypeS('TXRTLBitIndex', 'Integer
7059: Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7060: Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7061: Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7062: Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7063: Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7064: Function XRTLSwapHiLo16( X : Word) : Word

```

```

7065: Function XRTLSwapHiLo32( X : Cardinal ) : Cardinal
7066: Function XRTLSwapHiLo64( X : Int64 ) : Int64
7067: Function XRTLROL32( A, S : Cardinal ) : Cardinal
7068: Function XRTLROL32( A, S : Cardinal ) : Cardinal
7069: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7070: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7071: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7072: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7073: //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7074: //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7075: Procedure XRTLUMul64( const A, B : Integer; var Mull, MulH : Integer)
7076: Function XRTLPopulation( A : Cardinal ) : Cardinal
7077: end;
7078:
7079: Function XRTLURLDecode( const ASrc : WideString ) : WideString
7080: Function XRTLURLEncode( const ASrc : WideString ) : WideString
7081: Function XRTLURINormalize( const AURI : WideString ) : WideString
7082: Procedure XRTLURIParse(const AURI:Widestring;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
VPassword : WideString)
7083: Function XRTLExtractLongPathName(APath: string): string;
7084:
7085: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7086: begin
7087:   AddTypeS('Int8', 'ShortInt
7088:   AddTypeS('Int16', 'SmallInt
7089:   AddTypeS('Int32', 'LongInt
7090:   AddTypeS('UInt8', 'Byte
7091:   AddTypeS('UInt16', 'Word
7092:   AddTypeS('UInt32', 'LongWord
7093:   AddTypeS('UInt64', 'Int64
7094:   AddTypeS('Word8', 'UInt8
7095:   AddTypeS('Word16', 'UInt16
7096:   AddTypeS('Word32', 'UInt32
7097:   AddTypeS('Word64', 'UInt64
7098:   AddTypeS('LargeInt', 'Int64
7099:   AddTypeS('NativeInt', 'Integer
7100:   AddTypeS('NativeUInt', 'Cardinal
7101:   Const('BitsPerByte','LongInt' ( 8);
7102:   Const('BitsPerWord','LongInt' ( 16);
7103:   Const('BitsPerLongWord','LongInt' ( 32);
7104:   //Const('BitsPerCardinal','LongInt' ( BytesPerCardinal * 8);
7105:   //Const('BitsPerNativeWord','LongInt' ( BytesPerNativeWord * 8);
7106:   Function MinI( const A, B : Integer ) : Integer
7107:   Function MaxI( const A, B : Integer ) : Integer
7108:   Function MinC( const A, B : Cardinal ) : Cardinal
7109:   Function MaxC( const A, B : Cardinal ) : Cardinal
7110:   Function SumClipI( const A, I : Integer ) : Integer
7111:   Function SumClipC( const A : Cardinal; const I : Integer ) : Cardinal
7112:   Function InByteRange( const A : Int64 ) : Boolean
7113:   Function InWordRange( const A : Int64 ) : Boolean
7114:   Function InLongWordRange( const A : Int64 ) : Boolean
7115:   Function InShortIntRange( const A : Int64 ) : Boolean
7116:   Function InSmallIntRange( const A : Int64 ) : Boolean
7117:   Function InLongIntRange( const A : Int64 ) : Boolean
7118:   AddTypeS('Bool8', 'ByteBool
7119:   AddTypeS('Bool16', 'WordBool
7120:   AddTypeS('Bool32', 'LongBool
7121:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7122:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7123:   Function ReverseCompareResult( const C : TCompareResult ) : TCompareResult
7124:   Const('MinSingle','Single').setExtended( 1.5E-45);
7125:   Const('MaxSingle','Single').setExtended( 3.4E+38);
7126:   Const('MinDouble','Double').setExtended( 5.0E-324);
7127:   Const('MaxDouble','Double').setExtended( 1.7E+308);
7128:   Const('MinExtended','Extended').setExtended(3.4E-4932);
7129:   Const('MaxExtended','Extended').setExtended(1.1E+4932);
7130:   Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7131:   Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7132:   Function MinF( const A, B : Float ) : Float
7133:   Function MaxF( const A, B : Float ) : Float
7134:   Function ClipF( const Value : Float; const Low, High : Float ) : Float
7135:   Function InSingleRange( const A : Float ) : Boolean
7136:   Function InDoubleRange( const A : Float ) : Boolean
7137:   Function InCurrencyRange( const A : Float ) : Boolean;
7138:   Function InCurrencyRange1( const A : Int64 ) : Boolean;
7139:   Function FloatExponentBase2( const A : Extended; var Exponent : Integer ) : Boolean
7140:   Function FloatExponentBase10( const A : Extended; var Exponent : Integer ) : Boolean
7141:   Function FloatIsInfinity( const A : Extended ) : Boolean
7142:   Function FloatIsNaN( const A : Extended ) : Boolean
7143:   Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7144:   Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7145:   Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7146:   Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7147:   Function FloatZero( const A : Float; const CompareDelta : Float ) : Boolean
7148:   Function FloatOne( const A : Float; const CompareDelta : Float ) : Boolean
7149:   Function FloatsEqual( const A, B : Float; const CompareDelta : Float ) : Boolean
7150:   Function FloatsCompare( const A, B : Float; const CompareDelta : Float ) : TCompareResult
7151:   Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7152:   Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);

```

```

7153: Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7154: Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7155: Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7156: Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7157: Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7158: Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7159: Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7160: Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7161: Function cIsHighBitSet( const Value : LongWord) : Boolean
7162: Function SetBitScanForward( const Value : LongWord) : Integer;
7163: Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7164: Function SetBitScanReverse( const Value : LongWord) : Integer;
7165: Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7166: Function ClearBitScanForward( const Value : LongWord) : Integer;
7167: Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7168: Function ClearBitScanReverse( const Value : LongWord) : Integer;
7169: Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7170: Function cReverseBits( const Value : LongWord) : LongWord;
7171: Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7172: Function cSwapEndian( const Value : LongWord) : LongWord
7173: Function cTwosComplement( const Value : LongWord) : LongWord
7174: Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7175: Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7176: Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7177: Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7178: Function cBitCount( const Value : LongWord) : LongWord
7179: Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7180: Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7181: Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7182: Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7183: Function SetBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord
7184: Function ClearBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord
7185: Function ToggleBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord
7186: Function IsBitRangeSet( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7187: Function IsBitRangeClear( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7188: // AddTypes('CharSet', 'set of AnsiChar
7189: AddTypes('CharSet', 'set of Char' //!!!
7190: AddTypes('AnsiCharSet', 'TCharSet
7191: AddTypes('ByteSet', 'set of Byte
7192: AddTypes('AnsiChar', 'Char
7193: // Function AsCharSet( const C : array of AnsiChar) : CharSet
7194: Function AsByteSet( const C : array of Byte) : ByteSet
7195: Procedure ComplementChar( var C : CharSet; const Ch : Char)
7196: Procedure ClearCharSet( var C : CharSet)
7197: Procedure FillCharSet( var C : CharSet)
7198: Procedure ComplementCharSet( var C : CharSet)
7199: Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7200: Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7201: Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7202: Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7203: Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7204: Function IsSubSet( const A, B : CharSet) : Boolean
7205: Function IsEqual( const A, B : CharSet) : Boolean
7206: Function IsEmpty( const C : CharSet) : Boolean
7207: Function IsComplete( const C : CharSet) : Boolean
7208: Function cCharCount( const C : CharSet) : Integer
7209: Procedure ConvertCaseInsensitive( var C : CharSet)
7210: Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7211: Function IntRangeLength( const Low, High : Integer) : Int64
7212: Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7213: Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7214: Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7215: Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7216: Function IntRangeIncludeElementRange( var Low, High : Integer; const LowElement, HighElement : Integer) : Boolean
7217: Function CardRangeLength( const Low, High : Cardinal) : Int64
7218: Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7219: Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7220: Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7221: Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7222: Function CardRangeIncludeElementRange( var Low, High : Cardinal; const LowElement, HighElement : Cardinal) : Boolean;
7223: AddTypes('UnicodeChar', 'WideChar
7224: Function Compare( const I1, I2 : Boolean) : TCompareResult;
7225: Function Compare1( const I1, I2 : Integer) : TCompareResult;
7226: Function Compare2( const I1, I2 : Int64) : TCompareResult;
7227: Function Compare3( const I1, I2 : Extended) : TCompareResult;
7228: Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7229: Function CompareW( const I1, I2 : WideString) : TCompareResult
7230: Function cSgn( const A : LongInt) : Integer;
7231: Function cSgn1( const A : Int64) : Integer;
7232: Function cSgn2( const A : Extended) : Integer;
7233: AddTypes('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7234: Function AnsiCharToInt( const A : AnsiChar) : Integer
7235: Function WideCharToInt( const A : WideChar) : Integer
7236: Function CharToInt( const A : Char) : Integer
7237: Function IntToAnsiChar( const A : Integer) : AnsiChar
7238: Function IntToWideChar( const A : Integer) : WideChar
7239: Function IntToChar( const A : Integer) : Char
7240: Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7241: Function IsHexWideChar( const Ch : WideChar) : Boolean

```



```

7242: Function IsHexChar( const Ch : Char ) : Boolean
7243: Function HexAnsiCharToInt( const A : AnsiChar ) : Integer
7244: Function HexWideCharToInt( const A : WideChar ) : Integer
7245: Function HexCharToInt( const A : Char ) : Integer
7246: Function IntToUpperHexAnsiChar( const A : Integer ) : AnsiChar
7247: Function IntToUpperHexWideChar( const A : Integer ) : WideChar
7248: Function IntToUpperHexChar( const A : Integer ) : Char
7249: Function IntToLowerHexAnsiChar( const A : Integer ) : AnsiChar
7250: Function IntToLowerHexWideChar( const A : Integer ) : WideChar
7251: Function IntToLowerHexChar( const A : Integer ) : Char
7252: Function IntToStringA( const A : Int64 ) : AnsiString
7253: Function IntToStringW( const A : Int64 ) : WideString
7254: Function IntToString( const A : Int64 ) : String
7255: Function UIntToStringA( const A : NativeUInt ) : AnsiString
7256: Function UIntToStringW( const A : NativeUInt ) : WideString
7257: Function UIntToString( const A : NativeUInt ) : String
7258: Function LongWordToStrA( const A : LongWord; const Digits : Integer ) : AnsiString
7259: Function LongWordToStrW( const A : LongWord; const Digits : Integer ) : WideString
7260: Function LongWordToStrU( const A : LongWord; const Digits : Integer ) : UnicodeString
7261: Function LongWordToStr( const A : LongWord; const Digits : Integer ) : String
7262: Function LongWordToHexA( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7263: Function LongWordToHexW( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7264: Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7265: Function LongWordToOctA( const A : LongWord; const Digits : Integer ) : AnsiString
7266: Function LongWordToOctW( const A : LongWord; const Digits : Integer ) : WideString
7267: Function LongWordToOct( const A : LongWord; const Digits : Integer ) : String
7268: Function LongWordToBinA( const A : LongWord; const Digits : Integer ) : AnsiString
7269: Function LongWordToBinW( const A : LongWord; const Digits : Integer ) : WideString
7270: Function LongWordToBin( const A : LongWord; const Digits : Integer ) : String
7271: Function TryStringToInt64A( const S : AnsiString; out A : Int64 ) : Boolean
7272: Function TryStringToInt64W( const S : WideString; out A : Int64 ) : Boolean
7273: Function TryStringToInt64( const S : String; out A : Int64 ) : Boolean
7274: Function StringToInt64DefA( const S : AnsiString; const Default : Int64 ) : Int64
7275: Function StringToInt64DefW( const S : WideString; const Default : Int64 ) : Int64
7276: Function StringToInt64Def( const S : String; const Default : Int64 ) : Int64
7277: Function StringToInt64A( const S : AnsiString ) : Int64
7278: Function StringToInt64W( const S : WideString ) : Int64
7279: Function StringToInt64( const S : String ) : Int64
7280: Function TryStringToIntA( const S : AnsiString; out A : Integer ) : Boolean
7281: Function TryStringToIntW( const S : WideString; out A : Integer ) : Boolean
7282: Function TryStringToInt( const S : String; out A : Integer ) : Boolean
7283: Function StringToIntDefA( const S : AnsiString; const Default : Integer ) : Integer
7284: Function StringToIntDefW( const S : WideString; const Default : Integer ) : Integer
7285: Function StringToIntDef( const S : String; const Default : Integer ) : Integer
7286: Function StringToIntA( const S : AnsiString ) : Integer
7287: Function StringToIntW( const S : WideString ) : Integer
7288: Function StringToInt( const S : String ) : Integer
7289: Function TryStringToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7290: Function TryStringToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7291: Function TryStringToLongWord( const S : String; out A : LongWord ) : Boolean
7292: Function StringToLongWordA( const S : AnsiString ) : LongWord
7293: Function StringToLongWordW( const S : WideString ) : LongWord
7294: Function StringToLongWord( const S : String ) : LongWord
7295: Function HexToUIntA( const S : AnsiString ) : NativeUInt
7296: Function HexToUIntW( const S : WideString ) : NativeUInt
7297: Function HexToUInt( const S : String ) : NativeUInt
7298: Function TryHexToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7299: Function TryHexToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7300: Function TryHexToLongWord( const S : String; out A : LongWord ) : Boolean
7301: Function HexToLongWordA( const S : AnsiString ) : LongWord
7302: Function HexToLongWordW( const S : WideString ) : LongWord
7303: Function HexToLongWord( const S : String ) : LongWord
7304: Function TryOctToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7305: Function TryOctToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7306: Function TryOctToLongWord( const S : String; out A : LongWord ) : Boolean
7307: Function OctToLongWordA( const S : AnsiString ) : LongWord
7308: Function OctToLongWordW( const S : WideString ) : LongWord
7309: Function OctToLongWord( const S : String ) : LongWord
7310: Function TryBinToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7311: Function TryBinToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7312: Function TryBinToLongWord( const S : String; out A : LongWord ) : Boolean
7313: Function BinToLongWordA( const S : AnsiString ) : LongWord
7314: Function BinToLongWordW( const S : WideString ) : LongWord
7315: Function BinToLongWord( const S : String ) : LongWord
7316: Function FloatToStringA( const A : Extended ) : AnsiString
7317: Function FloatToStringW( const A : Extended ) : WideString
7318: Function FloatToString( const A : Extended ) : String
7319: Function TryStringToFloatA( const A : AnsiString; out B : Extended ) : Boolean
7320: Function TryStringToFloatW( const A : WideString; out B : Extended ) : Boolean
7321: Function TryStringToFloat( const A : String; out B : Extended ) : Boolean
7322: Function StringToFloatA( const A : AnsiString ) : Extended
7323: Function StringToFloatW( const A : WideString ) : Extended
7324: Function StringToFloat( const A : String ) : Extended
7325: Function StringToFloatDefA( const A : AnsiString; const Default : Extended ) : Extended
7326: Function StringToFloatDefW( const A : WideString; const Default : Extended ) : Extended
7327: Function StringToFloatDef( const A : String; const Default : Extended ) : Extended
7328: Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean; const PadMultiple:Integer; const
PadChar: AnsiChar) : AnsiString
7329: Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet ) : AnsiString

```

```

7330: unit uPSI_cFundamentUtils;
7331: Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7332: Const('b64_UUEncode','String').String('!\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7333: Const('b64_XXEncode','String').String('+~0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7334: Const('CCHARSET','Stringb64_XXEncode');
7335: Const('CHEXSET','String'0123456789ABCDEF
7336: Const('HEXDIGITS','String'0123456789ABCDEF
7337: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7338: Const('DIGISET','String'0123456789
7339: Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7340: Const('DIGISET2','TCharset').SetSet('0123456789'
7341: Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7342: Const('HEXSET2','TCharset').SetSet('0123456789ABCDEF');
7343: Const('NUMBERSET','TCharset').SetSet('0123456789');
7344: Const('NUMBERS','String'0123456789');
7345: Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7346: Function CharSetToStr( const C : CharSet) : AnsiString
7347: Function StrToCharSet( const S : AnsiString) : CharSet
7348: Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7349: Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7350: Function UUDecode( const S : AnsiString) : AnsiString
7351: Function XXDecode( const S : AnsiString) : AnsiString
7352: Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7353: Function InterfaceToStrA( const I : IInterface) : AnsiString
7354: Function InterfaceToStrW( const I : IInterface) : WideString
7355: Function InterfaceToStr( const I : IInterface) : String
7356: Function ObjectClassName( const O : TObject) : String
7357: Function ClassClassName( const C : TClass) : String
7358: Function ObjectToStr( const O : TObject) : String
7359: Function ObjectToString( const O : TObject) : String
7360: Function CharSetToStr( const C : CharSet) : AnsiString
7361: Function StrToCharSet( const S : AnsiString) : CharSet
7362: Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
  AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7363: Function HashStrW(const S:Widestring;const Index:Integer;const Count:Integer;const
  AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7364: Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
  : Boolean; const Slots : LongWord) : LongWord
7365: Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7366: Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7367: Const('Bytes1KB','LongInt' ( 1024);
7368: SIRegister_IInterface(CL);
7369: Procedure SelfTestCFundamentUtils
7370:
7371: Function CreateSchedule : IJclSchedule
7372: Function NullStamp : TTimeStamp
7373: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7374: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7375: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7376:
7377: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7378: begin
7379:   AddTypeS('TFunc', 'function(X : Float) : Float;
7380: Function InitGraphics( Width, Height : Integer) : Boolean
7381: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7382: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7383: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7384: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7385: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7386: Procedure SetGraphTitle( Title : String)
7387: Procedure SetOxTitle( Title : String)
7388: Procedure SetOyTitle( Title : String)
7389: Function GetGraphTitle : String
7390: Function GetOxTitle : String
7391: Function GetOyTitle : String
7392: Procedure PlotOxAxis( Canvas : TCanvas)
7393: Procedure PlotOyAxis( Canvas : TCanvas)
7394: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7395: Procedure WriteGraphTitle( Canvas : TCanvas)
7396: Function SetMaxCurv( NCurv : Byte) : Boolean
7397: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7398: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7399: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7400: Procedure SetCurvStep( CurvIndex, Step : Integer)
7401: Function GetMaxCurv : Byte
7402: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7403: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7404: Function GetCurvLegend( CurvIndex : Integer) : String
7405: Function GetCurvStep( CurvIndex : Integer) : Integer
7406: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7407: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7408: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7409: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7410: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7411: Procedure ConRec( Cnvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7412: Function Xpixel( X : Float) : Integer
7413: Function Ypixel( Y : Float) : Integer
7414: Function Xuser( X : Integer) : Float
7415: Function Yuser( Y : Integer) : Float

```

```

7416: end;
7417:
7418: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7419: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7420: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7421: Procedure FFT_Integer_Cleanup
7422: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7423: //unit uPSI_JclStreams;
7424: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7425: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7426: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7427: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7428:
7429: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7430: begin
7431:   FindClass('TOBJECT'),'EInvalidDest
7432:   FindClass('TOBJECT'),'EFCantMove
7433:   Procedure fmxCopyFile( const FileName, DestName : string)
7434:   Procedure fmxMoveFile( const FileName, DestName : string)
7435:   Function fmxGetFileSize( const FileName : string) : LongInt
7436:   Function fmxFileDateTime( const FileName : string) : TDateTime
7437:   Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7438:   Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7439: end;
7440:
7441: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7442: begin
7443:   SIRegister_IFindFileIterator(CL);
7444:   Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7445: end;
7446:
7447: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7448: begin
7449:   Function SkipWhite( cp : PChar) : PChar
7450:   Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7451:   Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7452:   Function ReadIdent( cp : PChar; var ident : string) : PChar
7453: end;
7454:
7455: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7456: begin
7457:   SIRegister_TStringHashMapTraits(CL);
7458:   Function CaseSensitiveTraits : TStringHashMapTraits
7459:   Function CaseInsensitiveTraits : TStringHashMapTraits
7460:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7461:   + 'e; Right : PHashNode; end
7462:   //PHashArray', 'THashArray // will not work
7463:   SIRegister_TStringHashMap(CL);
7464:   THashValue', 'Cardinal
7465:   Function StrHash( const s : string) : THashValue
7466:   Function TextHash( const s : string) : THashValue
7467:   Function DataHash( var AValue, ASize : Cardinal) : THashValue
7468:   Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7469:   Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7470:   Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7471:   SIRegister_TCaseSensitiveTraits(CL);
7472:   SIRegister_TCaseInsensitiveTraits(CL);
7473:
7474:
7475: //*****unit uPSI_umath;
7476: Function uExpo( X : Float) : Float
7477: Function uExp2( X : Float) : Float
7478: Function uExp10( X : Float) : Float
7479: Function uLog( X : Float) : Float
7480: Function uLog2( X : Float) : Float
7481: Function uLog10( X : Float) : Float
7482: Function uLogA( X, A : Float) : Float
7483: Function uIntPower( X : Float; N : Integer): Float
7484: Function uPower( X, Y : Float) : Float
7485: Function SgnGamma( X : Float) : Integer
7486: Function Stirling( X : Float) : Float
7487: Function StirLog( X : Float) : Float
7488: Function Gamma( X : Float) : Float
7489: Function LnGamma( X : Float) : Float
7490: Function DiGamma( X : Float) : Float
7491: Function TriGamma( X : Float) : Float
7492: Function IGamma( X : Float) : Float
7493: Function JGamma( X : Float) : Float
7494: Function InvGamma( X : Float) : Float
7495: Function Erf( X : Float) : Float
7496: Function Erfc( X : Float) : Float
7497: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7498: { Correlation coefficient between samples X and Y }
7499: function DBeta(A, B, X : Float) : Float;
7500: { Density of Beta distribution with parameters A and B }
7501: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7502: Function Beta(X, Y : Float) : Float
7503: Function Binomial( N, K : Integer) : Float
7504: Function PBinom( N : Integer; P : Float; K : Integer) : Float

```

```

7505: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7506: Procedure LU_Decom( A : TMatrix; Lb, Ub : Integer)
7507: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7508: Function DNorm( X : Float) : Float
7509:
7510: function DGamma(A, B, X : Float) : Float;
7511: { Density of Gamma distribution with parameters A and B }
7512: function DKhi2(Nu : Integer; X : Float) : Float;
7513: { Density of Khi-2 distribution with Nu d.o.f. }
7514: function DStudent(Nu : Integer; X : Float) : Float;
7515: { Density of Student distribution with Nu d.o.f. }
7516: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7517: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7518: function IBeta(A, B, X : Float) : Float;
7519: { Incomplete Beta function }
7520: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7521:
7522: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7523: begin
7524:   Procedure SetOptAlgo( Algo : TOptAlgo)
7525: procedure SetOptAlgo(Algo : TOptAlgo);
7526: { -----
7527:   Sets the optimization algorithm according to Algo, which must be
7528:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7529:
7530: Function GetOptAlgo : TOptAlgo
7531: Procedure SetMaxParam( N : Byte)
7532: Function GetMaxParam : Byte
7533: Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7534: Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7535: Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7536: Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7537: Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7538: Procedure SetMCFile( FileName : String)
7539: Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7540: Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
LastPar:Integer;V:TMatrix);
7541: end;
7542:
7543: (*-----*)
7544: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7545: begin
7546:   Procedure SaveSimplex( FileName : string)
7547:   Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7548: end;
7549: (*-----*)
7550: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7551: begin
7552:   Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7553:   Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7554: end;
7555:
7556: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7557: begin
7558:   Function LTrim( S : String) : String
7559:   Function RTrim( S : String) : String
7560:   Function uTrim( S : String) : String
7561:   Function StrChar( N : Byte; C : Char) : String
7562:   Function RFill( S : String; L : Byte) : String
7563:   Function LFill( S : String; L : Byte) : String
7564:   Function CFill( S : String; L : Byte) : String
7565:   Function Replace( S : String; C1, C2 : Char) : String
7566:   Function Extract( S : String; var Index : Byte; Delim : Char) : String
7567:   Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7568:   Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7569:   Function FloatStr( X : Float) : String
7570:   Function IntStr( N : LongInt) : String
7571:   Function uCompStr( Z : Complex) : String
7572: end;
7573:
7574: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7575: begin
7576:   Function uSinh( X : Float) : Float
7577:   Function uCosh( X : Float) : Float
7578:   Function uTanh( X : Float) : Float
7579:   Function uArcSinh( X : Float) : Float
7580:   Function uArcCosh( X : Float) : Float
7581:   Function ArcTanh( X : Float) : Float
7582:   Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7583: end;
7584:
7585: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7586: begin
7587: type RNG_Type =
7588:   (RNG_MWC,      { Multiply-With-Carry }
7589:   RNG_MT,        { Mersenne Twister }
7590:   RNG_UVAG);    { Universal Virtual Array Generator }

```



```

7591: Procedure SetRNG( RNG : RNG_Type)
7592: Procedure InitGen( Seed : RNG_IntType)
7593: Procedure SRand( Seed : RNG_IntType)
7594: Function IRanGen : RNG_IntType
7595: Function IRanGen31 : RNG_IntType
7596: Function RanGen1 : Float
7597: Function RanGen2 : Float
7598: Function RanGen3 : Float
7599: Function RanGen53 : Float
7600: end;
7601:
7602: // Optimization by Simulated Annealing
7603: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7604: begin
7605:   Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7606:   Procedure SA_CreateLogFile( FileName : String)
7607:   Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7608: end;
7609:
7610: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7611: begin
7612:   Procedure InitUVAGbyString( KeyPhrase : string)
7613:   Procedure InitUVAG( Seed : RNG_IntType)
7614:   Function IRanUVAG : RNG_IntType
7615: end;
7616:
7617: procedure SIRegister_uenalg(CL: TPSPascalCompiler);
7618: begin
7619:   Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7620:   Procedure GA_CreateLogFile( LogFileName : String)
7621:   Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7622: end;
7623:
7624: TVector', 'array of Float
7625: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7626: begin
7627:   Procedure QSort( X : TVector; Lb, Ub : Integer)
7628:   Procedure DQSort( X : TVector; Lb, Ub : Integer)
7629: end;
7630:
7631: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7632: begin
7633:   Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7634:   Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7635: end;
7636:
7637: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7638: begin
7639:   FT_Result', 'Integer
7640:   //TDWordptr', '^DWord // will not work
7641:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor
7642:   d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha
7643:   r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP
7644:   ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeUp : Word; Rev4 : B
7645:   yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By
7646:   te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte
7647:   ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S
7648:   erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh
7649:   Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By
7650:   te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B
7651:   yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs :
7652:   Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I
7653:   nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv
7654:   ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0
7655:   : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B
7656:   yte; end
7657: end;
7658:
7659:
7660: //***** PaintFX*****
7661: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7662: begin
7663:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7664:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7665:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7666:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7667:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7668:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7669:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7670:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7671:     Procedure Turn( Src, Dst : TBitmap)
7672:     Procedure TurnRight( Src, Dst : TBitmap)
7673:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7674:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7675:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7676:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7677:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7678:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7679:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)

```

```

7680: Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7681: Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7682: Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7683: Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7684: Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7685: Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7686: Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7687: Procedure Emboss( var Bmp : TBitmap)
7688: Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7689: Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7690: Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7691: Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7692: Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7693: Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7694: Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7695: Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7696: Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7697: Procedure QuartoOpaque( Src, Dst : TBitmap)
7698: Procedure SemiOpaque( Src, Dst : TBitmap)
7699: Procedure ShadowDownLeft( const Dst : TBitmap)
7700: Procedure ShadowDownRight( const Dst : TBitmap)
7701: Procedure ShadowUpLeft( const Dst : TBitmap)
7702: Procedure ShadowUpRight( const Dst : TBitmap)
7703: Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7704: Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7705: Procedure FlipRight( const Dst : TBitmap)
7706: Procedure FlipDown( const Dst : TBitmap)
7707: Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7708: Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7709: Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7710: Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7711: Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7712: Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7713: Procedure SmoothResize( var Src, Dst : TBitmap)
7714: Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7715: Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7716: Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7717: Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7718: Procedure GrayScale( const Dst : TBitmap)
7719: Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7720: Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7721: Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7722: Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7723: Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7724: Procedure Spray( const Dst : TBitmap; Amount : Integer)
7725: Procedure AntiAlias( const Dst : TBitmap)
7726: Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7727: Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7728: Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7729: Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7730: Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7731: Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7732: Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7733: Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7734: Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7735: Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7736: Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7737: Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7738: Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7739: Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7740: Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7741: Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7742: Procedure Invert( Src : TBitmap)
7743: Procedure MirrorRight( Src : TBitmap)
7744: Procedure MirrorDown( Src : TBitmap)
7745: end;
7746: end;
7747:
7748: (*-----*)
7749: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7750: begin
7751:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7752:     + 'ye, lbrotate, lbtwist, lbwrist, lbbrimble, mbHor, mbBottom, mbDiamond, mbWast'
7753:     + 'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )'
7754:   SIRegister_TJvPaintFX(CL);
7755:   Function SplineFilter( Value : Single) : Single
7756:   Function BellFilter( Value : Single) : Single
7757:   Function TriangleFilter( Value : Single) : Single
7758:   Function BoxFilter( Value : Single) : Single
7759:   Function HermiteFilter( Value : Single) : Single
7760:   Function Lanczos3Filter( Value : Single) : Single
7761:   Function MitchellFilter( Value : Single) : Single
7762: end;
7763:
7764:
7765: (*-----*)
7766: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7767: begin
7768:   'TeeMsg_DefaultFunctionName', 'String ' TeeFunction

```

```

7769: TeeMsg_DefaultSeriesName', 'String' Series
7770: TeeMsg_DefaultToolName', 'String' ChartTool
7771: ChartComponentPalette', 'String' TeeChart
7772: TeeMaxLegendColumns', 'LongInt' ( 2 );
7773: TeeDefaultLegendSymbolWidth', 'LongInt' ( 20 );
7774: TeeTitleFootDistance, 'LongInt' ( 5 );
7775: SIRegister_TCustomChartWall(CL);
7776: SIRegister_TChartWall(CL);
7777: SIRegister_TChartLegendGradient(CL);
7778: TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7779: TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7780: FindClass('TOBJECT'), 'LegendException
7781: TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7782:   + 'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7783: FindClass('TOBJECT'), 'TCustomChartLegend
7784: TLegendSymbolSize', '( lcsPercent, lcsPixels )
7785: TLegendSymbolPosition', '( spLeft, spRight )
7786: TSymbolDrawEvent', 'Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7787: TSymbolCalcHeight', 'Function : Integer
7788: SIRegister_TLegendSymbol(CL);
7789: SIRegister_TTeeCustomShapePosition(CL);
7790: TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7791: SIRegister_TLegendTitle(CL);
7792: SIRegister_TLegendItem(CL);
7793: SIRegister_TLegendItems(CL);
7794: TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7795: FindClass('TOBJECT'), 'TCustomChart
7796: SIRegister_TCustomChartLegend(CL);
7797: SIRegister_TChartLegend(CL);
7798: SIRegister_TChartTitle(CL);
7799: SIRegister_TChartFootTitle(CL);
7800: TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7801:   + 'eButton; Shift : TShiftState; X, Y : Integer)
7802: TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7803:   + 'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7804: TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series : '
7805:   + 'TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7806: TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7807:   + 'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7808: TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7809: TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7810: TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7811:   + 'toMax : Boolean; Min : Double; Max : Double; end
7812: TAllAxisSavedScales', 'array of TAxisSavedScales
7813: SIRegister_TChartBackWall(CL);
7814: SIRegister_TChartRightWall(CL);
7815: SIRegister_TChartBottomWall(CL);
7816: SIRegister_TChartLeftWall(CL);
7817: SIRegister_TChartWalls(CL);
7818: TChartAllowScrollEvent', 'Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7819: SIRegister_TCustomChart(CL);
7820: SIRegister_TChart(CL);
7821: SIRegister_TTeeSeriesTypes(CL);
7822: SIRegister_TTeeToolTypes(CL);
7823: SIRegister_TTeeDragObject(CL);
7824: SIRegister_TColorPalettes(CL);
7825: Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
AGalleryPage:PString;ANumGallerySeries:Integer;
7826: Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7827: Procedure RegisterTeeFunction(AFuncClass:T TeeFunctionClass;ADescription,
AGalleryPage:PString;ANumGallerySeries: Int;
7828: Procedure RegisterTeeBasicFunction( AFunctionClass : T TeeFunctionClass; ADescription : PString)
7829: Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:T TeeFunctionClass;
ADescription, AGalleryPage: PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7830: Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7831: Procedure UnRegisterTeeFunctions( const AFunctionList : array of T TeeFunctionClass)
7832: Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7833: Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : T TeeFunctionClass ) : T TeeFunction
7834: Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
AFunctionClass : T TeeFunctionClass ) : TChartSeries
7835: Function CloneChartSeries( ASeries : TChartSeries ) : TChartSeries;
7836: Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel ) : TChartSeries;
7837: Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7838: Function CloneChartTool( ATool : T TeeCustomTool; AOwner : TComponent ) : T TeeCustomTool
7839: Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass ) : TChartSeries
7840: Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7841: Function GetNewSeriesName( AOwner : TComponent ) : TComponentName
7842: Procedure RegisterTeeTools( const ATools : array of T TeeCustomToolClass)
7843: Procedure UnRegisterTeeTools( const ATools : array of T TeeCustomToolClass)
7844: Function GetGallerySeriesName( ASeries : TChartSeries ) : String
7845: Procedure PaintSeriesLegend(ASeries:TChartSeries; ACanvas:TCanvas; const R:TRect;ReferenceChart:
TCustomChart);
7846: SIRegister_TChartTheme(CL);
7847: //TChartThemeClass', 'class of TChartTheme
7848: //TCanvasClass', 'class of TCanvas3D
7849: Function SeriesNameOrIndex( ASeries : TCustomChartSeries ) : String
7850: Function SeriesTitleOrName( ASeries : TCustomChartSeries ) : String
7851: Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7852: Procedure ShowMessageUser( const S : String)

```

```

7853: Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7854: Function HasLabels( ASeries : TChartSeries) : Boolean
7855: Function HasColors( ASeries : TChartSeries) : Boolean
7856: Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7857: Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7858: end;
7859:
7860:
7861: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7862: begin
7863:   ('//TeeFormBorderStyle', ' bsNone');
7864:   SIRegister_TMtafile(CL);
7865:   'TeeDefVerticalMargin', 'LongInt'( 4);
7866:   'TeeDefHorizMargin', 'LongInt'( 3);
7867:   'crTeeHand', 'LongInt'( TCursor ( 2020 ));
7868:   'TeeMsg_TeeHand', 'String' 'crTeeHand
7869:   'TeeNormalPrintDetail', 'LongInt'( 0);
7870:   'TeeHighPrintDetail', 'LongInt'( - 100);
7871:   'TeeDefault_PrintMargin', 'LongInt'( 15);
7872:   'MaxDefaultColors', 'LongInt'( 19);
7873:   'TeeTabDelimiter', 'Char' #9);
7874:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7875:   + 'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7876:   + 'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7877:   + 'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7878:   + 'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7879:   + 'ourMonths, dtSixMonths, dtOneYear, dtNone )
7880:   SIRegister_TCustomPanelNoCaption(CL);
7881:   FindClass('TObject'), 'TCustomTeePanel
7882:   SIRegister_TZoomPanning(CL);
7883:   SIRegister_TTeeEvent(CL);
7884:   ('//SIRegister_TTeeEventListeners(CL);
7885:   'TeeMouseEventKind', '( meDown, meUp, meMove )
7886:   SIRegister_TTeeMouseEvent(CL);
7887:   SIRegister_TCustomTeePanel(CL);
7888:   ('//TChartGradient', 'TeeGradient
7889:   ('//TChartGradientClass', 'class of TChartGradient
7890:   'TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7891:   SIRegister_TTeeZoomPen(CL);
7892:   SIRegister_TTeeZoomBrush(CL);
7893:   'TeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7894:   SIRegister_TTeeZoom(CL);
7895:   FindClass('TObject'), 'TCustomTeePanelExtended
7896:   'TeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7897:   SIRegister_TBackImage(CL);
7898:   SIRegister_TCustomTeePanelExtended(CL);
7899:   ('//TChartBrushClass', 'class of TChartBrush
7900:   SIRegister_TTeeCustomShapeBrushPen(CL);
7901:   'TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7902:   'TTextFormat', '( ttfNormal, ttfHtml )
7903:   SIRegister_TTeeCustomShape(CL);
7904:   SIRegister_TTeeShape(CL);
7905:   SIRegister_TTeeExportData(CL);
7906:   Function TeeStr( const Num : Integer) : String
7907:   Function DateTimeDefaultFormat( const AStep : Double) : String
7908:   Function TEEDaysInMonth( Year, Month : Word) : Word
7909:   Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7910:   Function NextDateTimeStep( const AStep : Double) : Double
7911:   Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7912:   Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7913:   Function PointInLine2( const P, FromPoint, ToPoint: TPoint; const TolerancePixels: Integer): Boolean;
7914:   Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels: Integer): Boolean;
7915:   Function PointInLineTolerance( const P: TPoint; const px, py, qx, qy, TolerancePixels: Integer): Boolean;
7916:   Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7917:   Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7918:   Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7919:   Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7920:   Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7921:   Function PointInEllipse1( const P: TPoint; Left, Top, Right, Bottom : Integer) : Boolean;
7922:   Function DelphiToLocalFormat( const Format : String) : String
7923:   Function LocalToDelphiFormat( const Format : String) : String
7924:   Procedure TeeEnableControls( Enable: Boolean; const ControlArray : array of TControl)
7925:   Function TeeRoundDate( const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7926:   Procedure TeeDateTimeIncrement( IsDateTime: Boolean; Increment: Boolean; var Value: Double; const
AnIncrement: Double; tmpWhichDateTime: TDateTimeStep)
7927:   'TeeSortCompare', 'Function ( a, b : Integer) : Integer
7928:   'TeeSortSwap', 'Procedure ( a, b : Integer)
7929:   Procedure TeeSort( StartIndex, EndIndex: Integer; CompareFunc: TTeeSortCompare; SwapFunc: TTeeSortSwap);
7930:   Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : String
7931:   Function TeeExtractField( St : String; Index : Integer) : String;
7932:   Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7933:   Function TeeNumFields( St : String) : Integer;
7934:   Function TeeNumFields1( const St, Separator : String) : Integer;
7935:   Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7936:   Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7937:   ('// TColorArray', 'array of TColor
7938:   Function GetDefaultColor( const Index : Integer) : TColor
7939:   Procedure SetDefaultColorPalette;
7940:   Procedure SetDefaultColorPalette1( const Palette : array of TColor);

```



```

7941: 'TeeCheckBoxSize','LongInt'( 11);
7942: Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
7943: Function TEEStrToFloatDef( const S : string; const Default : Extended ) : Extended
7944: Function TryStrToFloat( const S : String; var Value : Double ) : Boolean
7945: Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double ) : Boolean
7946: Procedure TeeTranslateControl( AControl : TControl);
7947: Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
7948: Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char ) : String
7949: //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
7950: Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean ) : TBitmap
7951: //Procedure DrawBevel(Canvas:TCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
7952: //Function ScreenRatio( ACanvas : TCanvas3D ) : Double
7953: Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean ) : Boolean
7954: Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
7955: Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer ) : Integer
7956: Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
7957: Function TeeReadStringOption( const AKey, DefaultValue : String ) : String
7958: Procedure TeeSaveStringOption( const AKey, Value : String)
7959: Function TeeDefaultXMLEncoding : String
7960: Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
7961: TeeWindowHandle, 'Integer
7962: Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
7963: Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
7964: Function HtmlTextExtent( ACanvas : TCanvas; const Text : String ) : TSize
7965: end;
7966:
7967:
7968: using mXBDEUtils
7969: *****
7970: Procedure SetAlias( aAlias, aDirectory : String)
7971: Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
Desired:Variant;Size:Byte);
7972: Function GetFileVersionNumber( const FileName : String ) : TVersionNo
7973: Procedure SetBDE( aPath, aNode, aValue : String)
7974: function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
7975: Function GetSystemDirectory( lpBuffer : string; uSize : UINT ) : UINT
7976: Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT ) : UINT
7977: Function GetTempPath( nBufferLength : DWORD; lpBuffer : string ) : DWORD
7978: Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string ) : DWORD
7979: Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
7980:
7981:
7982: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
7983: begin
7984: AddClassN(FindClass('TOBJECT'),'EDateTime
7985: Function DatePart( const D : TDateTime ) : Integer
7986: Function TimePart( const D : TDateTime ) : Double
7987: Function Century( const D : TDateTime ) : Word
7988: Function Year( const D : TDateTime ) : Word
7989: Function Month( const D : TDateTime ) : Word
7990: Function Day( const D : TDateTime ) : Word
7991: Function Hour( const D : TDateTime ) : Word
7992: Function Minute( const D : TDateTime ) : Word
7993: Function Second( const D : TDateTime ) : Word
7994: Function Millisecond( const D : TDateTime ) : Word
7995: ('OneDay','Extended').setExtended( 1.0);
7996: ('OneHour','Extended').SetExtended( OneDay / 24);
7997: ('OneMinute','Extended').SetExtended( OneHour / 60);
7998: ('OneSecond','Extended').SetExtended( OneMinute / 60);
7999: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8000: ('OneWeek','Extended').SetExtended( OneDay * 7);
8001: ('HoursPerDay','Extended').SetExtended( 24);
8002: ('MinutesPerHour','Extended').SetExtended( 60);
8003: ('SecondsPerMinute','Extended').SetExtended( 60);
8004: Procedure SetYear( var D : TDateTime; const Year : Word)
8005: Procedure SetMonth( var D : TDateTime; const Month : Word)
8006: Procedure SetDay( var D : TDateTime; const Day : Word)
8007: Procedure SetHour( var D : TDateTime; const Hour : Word)
8008: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8009: Procedure SetSecond( var D : TDateTime; const Second : Word)
8010: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8011: Function IsEqual( const D1, D2 : TDateTime ) : Boolean;
8012: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8013: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8014: Function IsAM( const D : TDateTime ) : Boolean
8015: Function IsPM( const D : TDateTime ) : Boolean
8016: Function IsMidnight( const D : TDateTime ) : Boolean
8017: Function IsNoon( const D : TDateTime ) : Boolean
8018: Function IsSunday( const D : TDateTime ) : Boolean
8019: Function IsMonday( const D : TDateTime ) : Boolean
8020: Function IsTuesday( const D : TDateTime ) : Boolean
8021: Function IsWednesday( const D : TDateTime ) : Boolean
8022: Function IsThursday( const D : TDateTime ) : Boolean
8023: Function IsFriday( const D : TDateTime ) : Boolean
8024: Function IsSaturday( const D : TDateTime ) : Boolean
8025: Function IsWeekend( const D : TDateTime ) : Boolean
8026: Function Noon( const D : TDateTime ) : TDateTime
8027: Function Midnight( const D : TDateTime ) : TDateTime
8028: Function FirstDayOfMonth( const D : TDateTime ) : TDateTime

```

```

8029: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8030: Function NextWorkday( const D : TDateTime) : TDateTime
8031: Function PreviousWorkday( const D : TDateTime) : TDateTime
8032: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8033: Function LastDayOfYear( const D : TDateTime) : TDateTime
8034: Function EasterSunday( const Year : Word) : TDateTime
8035: Function GoodFriday( const Year : Word) : TDateTime
8036: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8037: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8038: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8039: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8040: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8041: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8042: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8043: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8044: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8045: Function DayOfYear( const D : TDateTime) : Integer
8046: Function DaysInMonth( const Ye, Mo : Word) : Integer
8047: Function DaysInMonth( const D : TDateTime) : Integer
8048: Function DaysInYear( const Ye : Word) : Integer
8049: Function DaysInYearDate( const D : TDateTime) : Integer
8050: Function WeekNumber( const D : TDateTime) : Integer
8051: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8052: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8053: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8054: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8055: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8056: Function DiffHours( const D1, D2 : TDateTime) : Integer
8057: Function DiffDays( const D1, D2 : TDateTime) : Integer
8058: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8059: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8060: Function DiffYears( const D1, D2 : TDateTime) : Integer
8061: Function GMTBias : Integer
8062: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8063: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8064: Function NowAsGMTTime : TDateTime
8065: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8066: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8067: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8068: Function DateTimeToANSI( const D : TDateTime) : Integer
8069: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8070: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8071: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8072: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8073: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8074: Function DateTimeAsElapsedTime( const D: TDateTime; const IncludeMilliseconds: Boolean): AnsiString
8075: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8076: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8077: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8078: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8079: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8080: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8081: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8082: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8083: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8084: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8085: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8086: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8087: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8088: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8089: Function EnglishShortMonthA( const S : AnsiString) : Integer
8090: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8091: Function EnglishLongMonthA( const S : AnsiString) : Integer
8092: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8093: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8094: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8095: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8096: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8097: Function RFCMonthA( const S : AnsiString) : Word
8098: Function RFCMonthU( const S : UnicodeString) : Word
8099: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds: Boolean) : AnsiString
8100: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds: Boolean) : UnicodeString
8101: Function GMTDateTimeToRFC1123DateTimeA( const D: TDateTime; const IncludeDayOfWeek: Bool): AnsiString;
8102: Function GMTDateTimeToRFC1123DateTimeU( const D: TDateTime; const IncludeDayOfWeek: Bool): UnicodeString;
8103: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8104: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8105: Function NowAsRFCDateTimeA : AnsiString
8106: Function NowAsRFCDateTimeU : UnicodeString
8107: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8108: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8109: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8110: Function TimePeriodStr( const D : TDateTime) : AnsiString
8111: Procedure SelfTest
8112: end;
8113: //*****CFileUtils
8114: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8115: Function PathHasDriveLetter( const Path : String) : Boolean
8116: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8117: Function PathIsDriveLetter( const Path : String) : Boolean

```

```

8118: Function PathIsDriveRootA( const Path : AnsiString ) : Boolean
8119: Function PathIsDriveRoot( const Path : String ) : Boolean
8120: Function PathIsRootA( const Path : AnsiString ) : Boolean
8121: Function PathIsRoot( const Path : String ) : Boolean
8122: Function PathIsUNCPathA( const Path : AnsiString ) : Boolean
8123: Function PathIsUNCPath( const Path : String ) : Boolean
8124: Function PathIsAbsoluteA( const Path : AnsiString ) : Boolean
8125: Function PathIsAbsolute( const Path : String ) : Boolean
8126: Function PathIsDirectoryA( const Path : AnsiString ) : Boolean
8127: Function PathIsDirectory( const Path : String ) : Boolean
8128: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8129: Function PathInclSuffix( const Path : String; const PathSep : Char ) : String
8130: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8131: Function PathExclSuffix( const Path : String; const PathSep : Char ) : String
8132: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char )
8133: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char )
8134: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char )
8135: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char )
8136: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8137: Function PathCanonical( const Path : String; const PathSep : Char ) : String
8138: Function PathExpandA( const Path:AnsiString; const BasePath:AnsiString; const PathSep:Char ):AnsiString
8139: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char ) : String
8140: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8141: Function PathLeftElement( const Path : String; const PathSep : Char ) : String
8142: Procedure PathSplitLeftElementA( const Path:AString; var LeftElement,RightPath:AString; const PathSep:Char );
8143: Procedure PathSplitLeftElement( const Path:String; var LeftElement,RightPath : String; const PathSep:Char );
8144: Procedure DecodeFilePathA( const FilePath:AnsiString; var Path,FileName:AnsiString; const PathSep:Char;
8145: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char )
8146: Function FileNameValidA( const FileName : AnsiString ) : AnsiString
8147: Function FileNameValid( const FileName : String ) : String
8148: Function FilePathA( const FileName,Path:AnsiString; const BasePath:AnsiStr; const PathSep:Char ):AnsiString;
8149: Function FilePath( const FileName, Path : String; const BasePath : String; const PathSep : Char ) : String
8150: Function DirectoryExpandA( const Path:AnsiString; const BasePath:AnsiString; const PathSep:Char ):AnsiString
8151: Function DirectoryExpand( const Path : String; const BasePath : String; const PathSep : Char ) : String
8152: Function UnixPathToWinPath( const Path : AnsiString ) : AnsiString
8153: Function WinPathToUnixPath( const Path : AnsiString ) : AnsiString
8154: Procedure CCopyFile( const FileName, DestName : String )
8155: Procedure CMoveFile( const FileName, DestName : String )
8156: Function CDeleteFiles( const FileMask : String ) : Boolean
8157: Function FileSeekEx( const FHandle:TFileHandle; const FileOffset:Int64; const FilePos:TFileSeekPos ):Int64;
8158: Procedure FileCloseEx( const FileHandle : TFileHandle )
8159: Function FileExistsA( const FileName : AnsiString ) : Boolean
8160: Function CFileExists( const FileName : String ) : Boolean
8161: Function CFileGetSize( const FileName : String ) : Int64
8162: Function FileGetDateTime( const FileName : String ) : TDateTime
8163: Function FileGetDateTime2( const FileName : String ) : TDateTime
8164: Function FileIsReadOnly( const FileName : String ) : Boolean
8165: Procedure FileDeleteEx( const FileName : String )
8166: Procedure FileRenameEx( const OldFileName, NewFileName : String )
8167: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
: TFileCreationMode; const FileOpenWait : PFileOpenWait ) : AnsiString
8168: Function DirectoryEntryExists( const Name : String ) : Boolean
8169: Function DirectoryEntrySize( const Name : String ) : Int64
8170: Function CDirectoryExists( const DirectoryName : String ) : Boolean
8171: Function DirectoryGetDateTime( const DirectoryName : String ) : TDateTime
8172: Procedure CDirectoryCreate( const DirectoryName : String )
8173: Function GetFirstFileNameMatching( const FileMask : String ) : String
8174: Function DirEntryGetAttr( const FileName : AnsiString ) : Integer
8175: Function DirEntryIsDirectory( const FileName : AnsiString ) : Boolean
8176: Function FileHasAttr( const FileName : String; const Attr : Word ) : Boolean
8177: AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8178: + 'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8179: Function DriveIsValid( const Drive : Char ) : Boolean
8180: Function DriveGetType( const Path : AnsiString ) : TLogicalDriveType
8181: Function DriveFreeSpace( const Path : AnsiString ) : Int64
8182:
8183: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8184: begin
8185:   AddClassN(FindClass('TOBJECT'),'ETimers
8186:   Const('TickFrequency','LongInt'( 1000));Function GetTick : LongWord
8187: Function TickDelta( const D1, D2 : LongWord ) : Integer
8188: Function TickDeltaW( const D1, D2 : LongWord ) : LongWord
8189:   AddTypeS('THPTimer', 'Int64
8190: Procedure StartTimer( var Timer : THPTimer )
8191: Procedure StopTimer( var Timer : THPTimer )
8192: Procedure ResumeTimer( var StoppedTimer : THPTimer )
8193: Procedure InitStoppedTimer( var Timer : THPTimer )
8194: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer )
8195: Function MillisecondsElapsed( const Timer : THPTimer; const TimerRunning : Boolean ) : Integer
8196: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean ) : Int64
8197: Procedure WaitMicroseconds( const MicroSeconds : Integer )
8198: Function GetHighPrecisionFrequency : Int64
8199: Function GetHighPrecisionTimerOverhead : Int64
8200: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64 )
8201: Procedure SelfTestCTimer
8202: end;
8203:
8204: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8205: begin

```

```

8206: Function RandomSeed : LongWord
8207: Procedure AddEntropy( const Value : LongWord)
8208: Function RandomUniform : LongWord;
8209: Function RandomUniform1( const N : Integer) : Integer;
8210: Function RandomBoolean : Boolean
8211: Function RandomByte : Byte
8212: Function RandomByteNonZero : Byte
8213: Function RandomWord : Word
8214: Function RandomInt64 : Int64;
8215: Function RandomInt64( const N : Int64) : Int64;
8216: Function RandomHex( const Digits : Integer) : String
8217: Function RandomFloat : Extended
8218: Function RandomAlphaStr( const Length : Integer) : AnsiString
8219: Function RandomPseudoWord( const Length : Integer) : AnsiString
8220: Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8221: Function mwcRandomLongWord : LongWord
8222: Function urnRandomLongWord : LongWord
8223: Function moaRandomFloat : Extended
8224: Function mwcRandomFloat : Extended
8225: Function RandomNormalF : Extended
8226: Procedure SelfTestCRandom
8227: end;
8228:
8229: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8230: begin
8231: // PIntArray', 'TIntArray // will not work
8232: Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8233: Addtypes('TConvertTabsProcEx','function(const Line:AnsiString; TabWidth: integer;var HasTabs: boolean):
AnsiString
8234: Function synMax( x, y : integer) : integer
8235: Function synMin( x, y : integer) : integer
8236: Function synMinMax( x, mi, ma : integer) : integer
8237: Procedure synSwapInt( var l, r : integer)
8238: Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8239: Function synMinPoint( const P1, P2 : TPoint) : TPoint
8240: //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8241: Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8242: Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8243: Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8244: Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8245: Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8246: Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8247: Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8248: Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8249: Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8250: Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8251: TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8252: + 'akana, stWideNumAlpha, stWideSymbol, stWideKatakana, stHiragana, stIdeograph, stControl, stKashida )
8253: ('C3_NONSPACING', 'LongInt'( 1);
8254: 'C3_DIACRITIC', 'LongInt'( 2);
8255: 'C3_VOWELMARK', 'LongInt'( 4);
8256: 'C3_SYMBOL', 'LongInt'( 8);
8257: 'C3_KATAKANA', 'LongWord( $0010);
8258: 'C3_HIRAGANA', 'LongWord( $0020);
8259: 'C3_HALFWIDTH', 'LongWord( $0040);
8260: 'C3_FULLWIDTH', 'LongWord( $0080);
8261: 'C3_IDEOGRAPH', 'LongWord( $0100);
8262: 'C3_KASHIDA', 'LongWord( $0200);
8263: 'C3_LEXICAL', 'LongWord( $0400);
8264: 'C3_ALPHA', 'LongWord( $8000);
8265: ('C3_NOTAPPLICABLE', 'LongInt'( 0);
8266: Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8267: Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8268: Function synIsStringType( Value : Word) : TStringType
8269: Function synGetEOL( Line : PChar) : PChar
8270: Function synEncodeString( s : string) : string
8271: Function synDecodeString( s : string) : string
8272: Procedure synFreeAndNil( var Obj: TObject)
8273: Procedure synAssert( Expr : Boolean)
8274: Function synLastDelimiter( const Delimiters, S : string) : Integer
8275: TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8276: TReplaceFlags', 'set of TReplaceFlag )
8277: Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8278: Function synGetRValue( RGBValue : TColor) : byte
8279: Function synGetGValue( RGBValue : TColor) : byte
8280: Function synGetBValue( RGBValue : TColor) : byte
8281: Function synRGB( r, g, b : Byte) : Cardinal
8282: // THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8283: // + 'lighter; Attr:TSynHighlighterAttributes;UniqueAttrName:string;Params array of Pointer):Boolean;
8284: //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8285: Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8286: Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8287: end;
8288:
8289: Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8290: Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8291: Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD

```



```

8292:
8293: procedure SIRegister_synutil(CL: TPSPascalCompiler);
8294: begin
8295:   Function STimeZoneBias : integer
8296:   Function TimeZone : string
8297:   Function Rfc822DateTime( t : TDateTime) : string
8298:   Function CDateTime( t : TDateTime) : string
8299:   Function SimpleDateTime( t : TDateTime) : string
8300:   Function AnsiCDateTime( t : TDateTime) : string
8301:   Function GetMonthNumber( Value : String) : integer
8302:   Function GetTimeFromStr( Value : string) : TDateTime
8303:   Function GetDateMDYFromStr( Value : string) : TDateTime
8304:   Function DecodeRfcDateTime( Value : string) : TDateTime
8305:   Function GetUTTime : TDateTime
8306:   Function SetUTTime( Newdt : TDateTime) : Boolean
8307:   Function SGetTick : LongWord
8308:   Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8309:   Function CodeInt( Value : Word) : Ansistring
8310:   Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8311:   Function CodeLongInt( Value : LongInt) : Ansistring
8312:   Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8313:   Function DumpStr( const Buffer : Ansistring) : string
8314:   Function DumpExStr( const Buffer : Ansistring) : string
8315:   Procedure Dump( const Buffer : Ansistring; DumpFile : string)
8316:   Procedure DumpEx( const Buffer : Ansistring; DumpFile : string)
8317:   Function TrimSPLeft( const S : string) : string
8318:   Function TrimSPRight( const S : string) : string
8319:   Function TrimSP( const S : string) : string
8320:   Function SeparateLeft( const Value, Delimiter : string) : string
8321:   Function SeparateRight( const Value, Delimiter : string) : string
8322:   Function SGetParameter( const Value, Parameter : string) : string
8323:   Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8324:   Procedure ParseParameters( Value : string; const Parameters : TStrings)
8325:   Function IndexByBegin( Value : string; const List : TStrings) : integer
8326:   Function GetEmailAddr( const Value : string) : string
8327:   Function GetEmailDesc( Value : string) : string
8328:   Function CStrToHex( const Value : Ansistring) : string
8329:   Function CIntToBin( Value : Integer; Digits : Byte) : string
8330:   Function CBinToInt( const Value : string) : Integer
8331:   Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8332:   Function CReplaceString( Value, Search, Replace : Ansistring) : Ansistring
8333:   Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8334:   Function CRPos( const Sub, Value : String) : Integer
8335:   Function FetchBin( var Value : string; const Delimiter : string) : string
8336:   Function CFetch( var Value : string; const Delimiter : string) : string
8337:   Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8338:   Function IsBinaryString( const Value : Ansistring) : Boolean
8339:   Function PosCRLF( const Value : Ansistring; var Terminator : Ansistring) : integer
8340:   Procedure StringsTrim( const value : TStrings)
8341:   Function PosFrom( const SubStr, Value : String; From : integer) : integer
8342:   Function IncPoint( const p : __pointer; Value : integer) : __pointer
8343:   Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8344:   Function CCountOfChar( const Value : string; aChr : char) : integer
8345:   Function UnquoteStr( const Value : string; Quote : Char) : string
8346:   Function QuoteStr( const Value : string; Quote : Char) : string
8347:   Procedure HeadersToList( const Value : TStrings)
8348:   Procedure ListToHeaders( const Value : TStrings)
8349:   Function SwapBytes( Value : integer) : integer
8350:   Function ReadStrFromStream( const Stream : TStream; len : integer) : Ansistring
8351:   Procedure WriteStrToStream( const Stream : TStream; Value : Ansistring)
8352:   Function GetTempFile( const Dir, prefix : Ansistring) : Ansistring
8353:   Function CPadString( const Value : Ansistring; len : integer; Pad : AnsiChar) : Ansistring
8354:   Function CXorString( Indatal, Indata2 : Ansistring) : Ansistring
8355:   Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8356: end;
8357:
8358: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8359: begin
8360:   ('CrcBufSize','LongInt'( 2048);
8361:   Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8362:   Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8363:   Function Adler32OfFile( FileName : Ansistring) : LongInt
8364:   Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8365:   Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8366:   Function Crc16OfFile( FileName : Ansistring) : Cardinal
8367:   Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8368:   Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8369:   Function Crc32OfFile( FileName : Ansistring) : LongInt
8370:   Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8371:   Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8372:   Function InternetSumOfFile( FileName : Ansistring) : Cardinal
8373:   Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8374:   Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8375:   Function Kermit16OfFile( FileName : Ansistring) : Cardinal
8376: end;
8377:
8378: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8379: begin
8380:   function CreateOleObject(const ClassName : String) : IDispatch;

```

```

8381: function GetActiveOleObject(const ClassName: String): IDispatch;
8382: function ProgIDToClassID(const ProgID: string): TGUID;
8383: function ClassIDToProgID(const ClassID: TGUID): string;
8384: function CreateClassID: string;
8385: function CreateGUIDString: string;
8386: function CreateGUIDID: string;
8387: procedure OleError(ErrorCode: longint);
8388: procedure OleCheck(Result: HRESULT);
8389: end;
8390:
8391: Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8392: Function xGetActiveOleObject( const ClassName : string) : Variant
8393: //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HRESULT
8394: Function DllCanUnloadNow : HRESULT
8395: Function DllRegisterServer : HRESULT
8396: Function DllUnregisterServer : HRESULT
8397: Function VarFromInterface( Unknown : IUnknown) : Variant
8398: Function VarToInterface( const V : Variant) : IDispatch
8399: Function VarToAutoObject( const V : Variant) : TAutoObject
8400: //Procedure
DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8401: //Procedure DispInvokeError( Status : HRESULT; const ExcepInfo : TExcepInfo)
8402: Procedure OleError( ErrorCode : HRESULT)
8403: Procedure OleCheck( Result : HRESULT)
8404: Function StringToClassID( const S : string) : TCLSID
8405: Function ClassIDToString( const ClassID : TCLSID) : string
8406: Function xProgIDToClassID( const ProgID : string) : TCLSID
8407: Function xClassIDToProgID( const ClassID : TCLSID) : string
8408: Function xWideCompareStr( const S1, S2 : WideString) : Integer
8409: Function xWideSameStr( const S1, S2 : WideString) : Boolean
8410: Function xGUIDToString( const ClassID : TGUID) : string
8411: Function xStringToGUID( const S : string) : TGUID
8412: Function xGetModuleName( Module : HMODULE) : string
8413: Function xAcquireExceptionObject : TObject
8414: Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8415: Function xUtf8Encode( const WS : WideString) : UTF8String
8416: Function xUtf8Decode( const S : UTF8String) : WideString
8417: Function xExcludeTrailingPathDelimiter( const S : string) : string
8418: Function xIncludeTrailingPathDelimiter( const S : string) : string
8419: Function XRTLHandleCOMException : HRESULT
8420: Procedure XRTLCheckArgument( Flag : Boolean)
8421: //Procedure XRTLCheckOutArgument( out Arg)
8422: Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
Connection:Longint);
8423: Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8424: Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
RegisterCookie:Int):HRESULT
8425: Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HRESULT
8426: //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HRESULT
8427: Procedure XRTLEnumActiveObjects( Strings : TStrings)
8428: function XRTLDefaultCategoryManager: IUnknown;
8429: function XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8430: // ICatRegister helper functions
8431: function XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
LocaleID: TLCID = LOCALE_USER_DEFAULT;
const CategoryManager: IUnknown = nil): HRESULT;
8432:
8433: function XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
LocaleID: TLCID = LOCALE_USER_DEFAULT;
const CategoryManager: IUnknown = nil): HRESULT;
8434:
8435: function XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
const CategoryManager: IUnknown = nil): HRESULT;
8436:
8437: function XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
const CategoryManager: IUnknown = nil): HRESULT;
8438:
8439: // ICatInformation helper functions
8440:
8441: function XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
LocaleID: TLCID = LOCALE_USER_DEFAULT;
const CategoryManager: IUnknown = nil): HRESULT;
8442:
8443: function XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
const CategoryManager: IUnknown = nil): HRESULT;
8444:
8445: function XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
const CategoryManager: IUnknown = nil): HRESULT;
8446:
8447: function XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
const CategoryManager: IUnknown = nil): HRESULT;
8448:
8449: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
const ADelete: Boolean = True): WideString;
8450:
8451: function XRTLFindPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8452:
8453: function XRTLGetVariantAsString( const Value : Variant) : string
8454:
8455: function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8456:
8457: function XRTLGetTimeZones : TXRTLTimeZones
8458:
8459: function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8460:
8461: function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8462:
8463: function GMTNow : TDateTime
8464:
8465: function GMTToLocalTime( GMT : TDateTime) : TDateTime
8466:
8467: function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8468:
8469: Procedure XRTLNotImplemented
8470:
8471: Procedure XRTLRaiseError( E : Exception)
8472:
8473: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8474:
8475:
8476:

```

```

8467: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8468: begin
8469:   SIRegister_IXRTLValue(CL);
8470:   SIRegister_TXRTLValue(CL);
8471:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8472:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8473: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8474: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8475: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8476: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8477: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8478: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8479: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8480: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8481: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8482: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8483: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8484: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8485: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8486: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8487: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8488: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8489: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8490: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8491: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8492: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8493: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8494: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8495: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8496: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8497: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8498: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8499: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8500: //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8501: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8502: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8503: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8504: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8505: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8506: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8507: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8508: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8509: Function XRTLGetAsObjectDef( const IValue:IXRTLValue;const DefValue:TObject;const
ADetachOwnership:Boolean):TObject;
8510: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8511: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8512: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8513: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8514: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8515: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8516: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8517: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8518: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8519: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8520: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8521: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8522: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8523: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8524: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8525: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8526: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8527: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8528: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8529: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8530: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8531: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8532: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8533: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8534: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8535: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8536: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8537: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8538: end;
8539:
8540: //*****unit uPSI_GR32;*****
8541:
8542: Function Color32( WinColor : TColor) : TColor32;
8543: Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8544: Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8545: Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8546: Function WinColor( Color32 : TColor32) : TColor
8547: Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8548: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8549: Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8550: Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8551: Function RedComponent( Color32 : TColor32) : Integer
8552: Function GreenComponent( Color32 : TColor32) : Integer
8553: Function BlueComponent( Color32 : TColor32) : Integer
8554: Function AlphaComponent( Color32 : TColor32) : Integer

```

```

8555: Function Intensity( Color32 : TColor32 ) : Integer
8556: Function SetAlpha( Color32 : TColor32; NewAlpha : Integer ) : TColor32
8557: Function HSLtoRGB( H, S, L : Single ) : TColor32;
8558: Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8559: Function HSLtoRGB1( H, S, L : Integer ) : TColor32;
8560: Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8561: Function WinPalette( const P : TPalette32 ) : HPALETTE
8562: Function FloatPoint( X, Y : Single ) : TFloatPoint;
8563: Function FloatPoint1( const P : TPoint ) : TFloatPoint;
8564: Function FloatPoint2( const FXP : TFixedPoint ) : TFloatPoint;
8565: Function FixedPoint( X, Y : Integer ) : TFixedPoint;
8566: Function FixedPoint1( X, Y : Single ) : TFixedPoint;
8567: Function FixedPoint2( const P : TPoint ) : TFixedPoint;
8568: Function FixedPoint3( const FP : TFloatPoint ) : TFixedPoint;
8569: AddTypes( 'TRectRounding', '( rrClosest, rrOutside, rrInside )
8570: Function MakeRect( const L, T, R, B : Integer ) : TRect;
8571: Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding ) : TRect;
8572: Function MakeRect2( const FXR : TRect; Rounding : TRectRounding ) : TRect;
8573: Function GFixedRect( const L, T, R, B : TFixed ) : TRect;
8574: Function FixedRect1( const ARect : TRect ) : TRect;
8575: Function FixedRect2( const FR : TFloatRect ) : TRect;
8576: Function GFloatRect( const L, T, R, B : TFloat ) : TFloatRect;
8577: Function FloatRect1( const ARect : TRect ) : TFloatRect;
8578: Function FloatRect2( const FXR : TRect ) : TFloatRect;
8579: Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect ) : Boolean;
8580: Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect ) : Boolean;
8581: Function GUnionRect( out Rect : TRect; const R1, R2 : TRect ) : Boolean;
8582: Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect ) : Boolean;
8583: Function GEqualRect( const R1, R2 : TRect ) : Boolean;
8584: Function EqualRect1( const R1, R2 : TFloatRect ) : Boolean;
8585: Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8586: Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8587: Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8588: Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8589: Function IsRectEmpty( const R : TRect ) : Boolean;
8590: Function IsRectEmpty1( const FR : TFloatRect ) : Boolean;
8591: Function GPtInRect( const R : TRect; const P : TPoint ) : Boolean;
8592: Function PtInRect1( const R : TFloatRect; const P : TPoint ) : Boolean;
8593: Function PtInRect2( const R : TRect; const P : TFloatPoint ) : Boolean;
8594: Function PtInRect3( const R : TFloatRect; const P : TFloatPoint ) : Boolean;
8595: Function EqualRectSize( const R1, R2 : TRect ) : Boolean;
8596: Function EqualRectSize1( const R1, R2 : TFloatRect ) : Boolean;
8597: Function MessageBeep( uType : UINT ) : BOOL
8598: Function ShowCursor( bShow : BOOL ) : Integer
8599: Function SetCursorPos( X, Y : Integer ) : BOOL
8600: Function SetCursor( hCursor : HICON ) : HCURSOR
8601: Function GetCursorPos( var lpPoint : TPoint ) : BOOL
8602: //Function ClipCursor( lpRect : PRect ) : BOOL
8603: Function GetClipCursor( var lpRect : TRect ) : BOOL
8604: Function GetCursor : HCURSOR
8605: Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer ) : BOOL
8606: Function GetCaretBlinkTime : UINT
8607: Function SetCaretBlinkTime( uMSeconds : UINT ) : BOOL
8608: Function DestroyCaret : BOOL
8609: Function HideCaret( hWnd : HWND ) : BOOL
8610: Function ShowCaret( hWnd : HWND ) : BOOL
8611: Function SetCaretPos( X, Y : Integer ) : BOOL
8612: Function GetCaretPos( var lpPoint : TPoint ) : BOOL
8613: Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint ) : BOOL
8614: Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint ) : BOOL
8615: Function MapWindowPoints( hWndFrom, hWndTo:HWND; var lpPoints, cPoints : UINT ) : Integer
8616: Function WindowFromPoint( Point : TPoint ) : HWND
8617: Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint ) : HWND
8618:
8619:
8620: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8621: begin
8622: Function FixedFloor( A : TFixed ) : Integer
8623: Function FixedCeil( A : TFixed ) : Integer
8624: Function FixedMul( A, B : TFixed ) : TFixed
8625: Function FixedDiv( A, B : TFixed ) : TFixed
8626: Function OneOver( Value : TFixed ) : TFixed
8627: Function FixedRound( A : TFixed ) : Integer
8628: Function FixedSqr( Value : TFixed ) : TFixed
8629: Function FixedSqrtLP( Value : TFixed ) : TFixed
8630: Function FixedSqrthp( Value : TFixed ) : TFixed
8631: Function FixedCombine( W, X, Y : TFixed ) : TFixed
8632: Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8633: Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8634: Function GRHypot( const X, Y : TFloat ) : TFloat;
8635: Function Hypot1( const X, Y : Integer ) : Integer;
8636: Function FastSqrt( const Value : TFloat ) : TFloat
8637: Function FastSqrtBab1( const Value : TFloat ) : TFloat
8638: Function FastSqrtBab2( const Value : TFloat ) : TFloat
8639: Function FastInvSqrt( const Value : Single ) : Single;
8640: Function MulDiv( Multiplicand, Multiplier, Divisor : Integer ) : Integer
8641: Function GRIsPowerOf2( Value : Integer ) : Boolean
8642: Function PrevPowerOf2( Value : Integer ) : Integer
8643: Function NextPowerOf2( Value : Integer ) : Integer

```



```

8644: Function Average( A, B : Integer) : Integer
8645: Function GRSign( Value : Integer) : Integer
8646: Function FloatMod( x, y : Double) : Double
8647: end;
8648:
8649: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8650: begin
8651:   Function Clamp( const Value : Integer) : Integer;
8652:   Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8653:   Function StackAlloc( Size : Integer) : Pointer
8654:   Procedure StackFree( P : Pointer)
8655:   Procedure Swap( var A, B : Pointer);
8656:   Procedure Swap1( var A, B : Integer);
8657:   Procedure Swap2( var A, B : TFixed);
8658:   Procedure Swap3( var A, B : TColor32);
8659:   Procedure TestSwap( var A, B : Integer);
8660:   Procedure TestSwap1( var A, B : TFixed);
8661:   Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8662:   Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8663:   Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8664:   Function Constrain1( const Value, Lo, Hi : Single) : Single;
8665:   Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8666:   Function GRMin( const A, B, C : Integer) : Integer;
8667:   Function GRMax( const A, B, C : Integer) : Integer;
8668:   Function Clamp( Value, Max : Integer) : Integer;
8669:   Function Clamp1( Value, Min, Max : Integer) : Integer;
8670:   Function Wrap( Value, Max : Integer) : Integer;
8671:   Function Wrap1( Value, Min, Max : Integer) : Integer;
8672:   Function Wrap3( Value, Max : Single) : Single;;
8673:   Function WrapPow2( Value, Max : Integer) : Integer;
8674:   Function WrapPow21( Value, Min, Max : Integer) : Integer;
8675:   Function Mirror( Value, Max : Integer) : Integer;
8676:   Function Mirror1( Value, Min, Max : Integer) : Integer;
8677:   Function MirrorPow2( Value, Max : Integer) : Integer;
8678:   Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8679:   Function GetOptimalWrap( Max : Integer) : TWrapProc;
8680:   Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8681:   Function GetOptimalMirror( Max : Integer) : TWrapProc;
8682:   Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8683:   Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8684:   Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8685:   Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8686:   Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8687:   Function Div255( Value : Cardinal) : Cardinal
8688:   Function SAR_4( Value : Integer) : Integer
8689:   Function SAR_8( Value : Integer) : Integer
8690:   Function SAR_9( Value : Integer) : Integer
8691:   Function SAR_11( Value : Integer) : Integer
8692:   Function SAR_12( Value : Integer) : Integer
8693:   Function SAR_13( Value : Integer) : Integer
8694:   Function SAR_14( Value : Integer) : Integer
8695:   Function SAR_15( Value : Integer) : Integer
8696:   Function SAR_16( Value : Integer) : Integer
8697:   Function ColorSwap( WinColor : TColor) : TColor32
8698: end;
8699:
8700: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8701: begin
8702:   AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8703:   Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8704:   Procedure CopyComponents1(Dst:TCustomBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8705:   Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8706:   Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8707:   Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8708:   Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8709:   Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8710:   Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8711:   Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8712:   Function CreateBitmap( Components : TColor32Components) : TColor32
8713:   Procedure ApplyBitmap(Dst:TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8714:   Procedure
      ApplyBitmap1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8715:   Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizedDst : Boolean)
8716: end;
8717:
8718:
8719: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8720: begin
8721:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8722:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8723:   Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8724:   Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8725:   Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8726:   Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8727:   Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8728:   Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8729:   Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;

```

```

8730: //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8731: //+'tedRangeBuffer; MoreData : Boolean; end
8732: Function NtfsSetSparse( const FileName : string) : Boolean
8733: Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8734: Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8735: //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
Ranges:TNtfsAllocRanges):Boolean;
8736: //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
Index:Integer):TFileAllocatedRangeBuffer
8737: Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8738: Function NtfsGetSparse( const FileName : string) : Boolean
8739: Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8740: Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8741: //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8742: Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8743: Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8744: Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8745: Function NtfsFolderMountPoint( const Path : string) : Boolean
8746: Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8747: Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8748: AddTypeS('TopLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8749: Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8750: Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8751: Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8752: Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8753: Function NtfsRequestOpLock( Handle : THandle; Kind : TopLock; Overlapped : TOverlapped) : Boolean
8754: Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8755: Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8756: Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8757: AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8758: + 'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8759: AddTypeS('TStreamIds', 'set of TStreamId
8760: AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8761: + ' : __Pointer; StreamIds : TStreamIds; end
8762: AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8763: + 'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8764: Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8765: Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8766: Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8767: Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8768: AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8769: Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8770: Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
List:TStrings):Bool;
8771: Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8772: Function JclAppInstances : TJclAppInstances;
8773: Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8774: Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8775: Procedure ReadMessageData( const Message : TMessage; var Data : __Pointer; var Size : Integer)
8776: Procedure ReadMessageString( const Message : TMessage; var S : string)
8777: Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8778:
8779:
8780: (*-----*)
8781: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8782: begin
8783: FindClass('TOBJECT'),'EJclGraphicsError
8784: TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8785: TDynPointArray', 'array of TPoint
8786: TDynDynPointArrayArray', 'array of TDynPointArray
8787: TPointF', 'record X : Single; Y : Single; end
8788: TDynPointArrayF', 'array of TPointF
8789: TDrawMode2', '( dmOpaque, dmBlend )
8790: TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8791: TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8792: TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8793: TMatrix3d', 'record array[0..2,0..2] of extended end
8794: TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8795: TScanLine', 'array of Integer
8796: TScanLines', 'array of TScanLine
8797: TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8798: TGradientDirection', '( gdVertical, gdHorizontal )
8799: TPolyFillMode', '( fmAlternate, fmWinding )
8800: TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8801: TJclRegionBitmapMode', '( rmInclude, rmExclude )
8802: TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8803: SIRegister_TJclDesktopCanvas(CL);
8804: FindClass('TOBJECT'),'TJclRegion
8805: SIRegister_TJclRegionInfo(CL);
8806: SIRegister_TJclRegion(CL);
8807: SIRegister_TJclThreadPersistent(CL);
8808: SIRegister_TJclCustomMap(CL);
8809: SIRegister_TJclBitmap32(CL);
8810: SIRegister_TJclByteMap(CL);
8811: SIRegister_TJclTransformation(CL);
8812: SIRegister_TJclLinearTransformation(CL);
8813: Procedure Stretch(NewWidth,
NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8814: Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);

```

```

8815: Procedure DrawBitmap( DC : HDC; Bitmap : HBitmap; X, Y, Width, Height : Integer)
8816: Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8817: Procedure BitmapToJpeg( const FileName : string)
8818: Procedure JpegToBitmap( const FileName : string)
8819: Function ExtractIconCount( const FileName : string) : Integer
8820: Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8821: Function IconToBitmapJ( Icon : HICON) : HBITMAP
8822: Procedure
BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8823: Procedure StretchTransfer(Dst:TJclBitmap32;
DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8824: Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8825: Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8826: Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
TGradientDirection) : Boolean;
8827: Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
RegionBitmapMode:TJclRegionBitmapMode): HRGN
8828: Procedure Screenshot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8829: Procedure Screenshot1( bm : TBitmap);
8830: Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8831: Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8832: Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8833: Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8834: Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8835: Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8836: Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8837: Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8838: Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8839: Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8840: Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8841: Procedure Invert( Dst, Src : TJclBitmap32)
8842: Procedure InvertRGB( Dst, Src : TJclBitmap32)
8843: Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8844: Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8845: Procedure SetGamma( Gamma : Single)
8846: end;
8847:
8848: (*-----*)
8849: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8850: begin
8851: Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8852: Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8853: Function LockedCompareExchange1( var Target : __Pointer; Exch, Comp : __Pointer) : Pointer;
8854: Function LockedDec( var Target : Integer) : Integer
8855: Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8856: Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8857: Function LockedExchangeDec( var Target : Integer) : Integer
8858: Function LockedExchangeInc( var Target : Integer) : Integer
8859: Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8860: Function LockedInc( var Target : Integer) : Integer
8861: Function LockedSub( var Target : Integer; Value : Integer) : Integer
8862: TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8863: SIRegister_TJclDispatcherObject(CL);
8864: Function WaitForMultipleObjects(const Objects:array of
TJclDispatcherObject;WaitAll:Bool;Timeout:Cardinal):Cardinal;
8865: Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
Timeout : Cardinal):Cardinal
8866: SIRegister_TJclCriticalSection(CL);
8867: SIRegister_TJclCriticalSectionEx(CL);
8868: SIRegister_TJclEvent(CL);
8869: SIRegister_TJclWaitableTimer(CL);
8870: SIRegister_TJclSemaphore(CL);
8871: SIRegister_TJclMutex(CL);
8872: POptexSharedInfo', '^TOptexSharedInfo // will not work
8873: TOptexSharedInfo', 'record SpinCount: Int; LockCount: Int; ThreadId: Longword; RecursionCount: Int; end
8874: SIRegister_TJclOptex(CL);
8875: TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8876: TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8877: TMrewThreadInfoArray', 'array of TMrewThreadInfo
8878: SIRegister_TJclMultiReadExclusiveWrite(CL);
8879: PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8880: TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
+ 'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8881: PMeteredSection', '^TMeteredSection // will not work
8882: TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8883: SIRegister_TJclMeteredSection(CL);
8884: TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8885: TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8886: TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount : Longint; end
8887: TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8888: Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8889: Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8890: Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8891: Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8892: Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8893: FindClass('TOBJECT'),'EJclWin32HandleObjectError
8894: FindClass('TOBJECT'),'EJclDispatcherObjectError
8895: FindClass('TOBJECT'),'EJclCriticalSectionError
8896: FindClass('TOBJECT'),'EJclEventError
8897: FindClass('TOBJECT'),'EJclEventError

```

```

8898: FindClass('TOBJECT'),'EJclWaitableTimerError
8899: FindClass('TOBJECT'),'EJclSemaphoreError
8900: FindClass('TOBJECT'),'EJclMutexError
8901: FindClass('TOBJECT'),'EJclMeteredSectionError
8902: end;
8903:
8904:
8905: //*****unit uPSI_mORMotReport;
8906: Procedure SetCurrentPrinterAsDefault
8907: Function CurrentPrinterName : string
8908: Function mCurrentPrinterPaperSize : string
8909: Procedure UseDefaultPrinter
8910:
8911: procedure SIRegisterTSTREAM(C1: TPSPascalCompiler);
8912: begin
8913:   with FindClass('TOBJECT'), 'TStream' do begin
8914:     IsAbstract := True;
8915:     //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8916:     // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8917:     function Read(Buffer:String;Count:LongInt):LongInt
8918:     function Write(Buffer:String;Count:LongInt):LongInt
8919:     function ReadString(Buffer:String;Count:LongInt):LongInt //FileStream
8920:     function WriteString(Buffer:String;Count:LongInt):LongInt
8921:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
8922:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
8923:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt);
8924:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt);
8925:
8926:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8927:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8928:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8929:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8930:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8931:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8932:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8933:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8934:
8935:     function Seek(Offset:LongInt;Origin:Word):LongInt
8936:     procedure ReadBuffer(Buffer:String;Count:LongInt)
8937:     procedure WriteBuffer(Buffer:String;Count:LongInt)
8938:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
8939:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)');
8940:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)');
8941:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)');
8942:
8943:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
8944:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
8945:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
8946:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
8947:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8948:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8949:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8950:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8951:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8952:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8953:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
8954:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
8955:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
8956:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
8957:
8958:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
8959:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
8960:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
8961:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)');
8962:     //READBUFFERAC
8963:     function InstanceSize: Longint
8964:     Procedure FixupResourceHeader( FixupInfo : Integer)
8965:     Procedure ReadResHeader
8966:
8967:     {$IFDEF DELPHI4UP}
8968:     function CopyFrom(Source:TStream;Count:Integer):LongInt
8969:     {$ELSE}
8970:     function CopyFrom(Source:TStream;Count:Integer):LongInt
8971:     {$ENDIF}
8972:     RegisterProperty('Position', 'LongInt', iptrw);
8973:     RegisterProperty('Size', 'LongInt', iptrw);
8974:   end;
8975: end;
8976:
8977:
8978: { *****
8979:   Unit DMATH - Interface for DMATH.DLL
8980:   ***** }
8981: // see more docs/dmath_manual.pdf
8982:
8983: Function InitEval : Integer
8984: Procedure SetVariable( VarName : Char; Value : Float)
8985: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
8986: Function Eval( ExpressionString : String) : Float

```



```

8987:
8988: unit dmath; //types are in built, others are external in DLL
8989: interface
8990: { $IFDEF DELPHI }
8991: uses
8992:   StdCtrls, Graphics;
8993: { $ENDIF }
8994: { -----
8995:   Types and constants
8996: ----- }
8997: { $i types.inc }
8998: { -----
8999:   Error handling
9000: ----- }
9001: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9002: { Sets the error code }
9003: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9004: { Sets error code and default function value }
9005: function MathErr : Integer; external 'dmath';
9006: { Returns the error code }
9007: { -----
9008:   Dynamic arrays
9009: ----- }
9010: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9011: { Sets the auto-initialization of arrays }
9012: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9013: { Creates floating point vector V[0..Ub] }
9014: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9015: { Creates integer vector V[0..Ub] }
9016: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9017: { Creates complex vector V[0..Ub] }
9018: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9019: { Creates boolean vector V[0..Ub] }
9020: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9021: { Creates string vector V[0..Ub] }
9022: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9023: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9024: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9025: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9026: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9027: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9028: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9029: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9030: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9031: { Creates string matrix A[0..Ub1, 0..Ub2] }
9032: { -----
9033:   Minimum, maximum, sign and exchange
9034: ----- }
9035: function FMin(X, Y : Float) : Float; external 'dmath';
9036: { Minimum of 2 reals }
9037: function FMax(X, Y : Float) : Float; external 'dmath';
9038: { Maximum of 2 reals }
9039: function IMin(X, Y : Integer) : Integer; external 'dmath';
9040: { Minimum of 2 integers }
9041: function IMax(X, Y : Integer) : Integer; external 'dmath';
9042: { Maximum of 2 integers }
9043: function Sgn(X : Float) : Integer; external 'dmath';
9044: { Sign (returns 1 if X = 0) }
9045: function Sgn0(X : Float) : Integer; external 'dmath';
9046: { Sign (returns 0 if X = 0) }
9047: function DSgn(A, B : Float) : Float; external 'dmath';
9048: { Sgn(B) * |A| }
9049: procedure FSwap(var X, Y : Float); external 'dmath';
9050: { Exchange 2 reals }
9051: procedure ISwap(var X, Y : Integer); external 'dmath';
9052: { Exchange 2 integers }
9053: { -----
9054:   Rounding functions
9055: ----- }
9056: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9057: { Rounds X to N decimal places }
9058: function Ceil(X : Float) : Integer; external 'dmath';
9059: { Ceiling function }
9060: function Floor(X : Float) : Integer; external 'dmath';
9061: { Floor function }
9062: { -----
9063:   Logarithms, exponentials and power
9064: ----- }
9065: function Expo(X : Float) : Float; external 'dmath';
9066: { Exponential }
9067: function Exp2(X : Float) : Float; external 'dmath';
9068: { 2^X }
9069: function Exp10(X : Float) : Float; external 'dmath';
9070: { 10^X }
9071: function Log(X : Float) : Float; external 'dmath';
9072: { Natural log }
9073: function Log2(X : Float) : Float; external 'dmath';
9074: { Log, base 2 }
9075: function Log10(X : Float) : Float; external 'dmath';

```

```

9076: { Decimal log }
9077: function LogA(X, A : Float) : Float; external 'dmath';
9078: { Log, base A }
9079: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9080: { X^N }
9081: function Power(X, Y : Float) : Float; external 'dmath';
9082: { X^Y, X >= 0 }
9083: { -----
9084:   Trigonometric functions
9085: ----- }
9086: function Pythag(X, Y : Float) : Float; external 'dmath';
9087: { Sqrt(X^2 + Y^2) }
9088: function FixAngle(Theta : Float) : Float; external 'dmath';
9089: { Set Theta in -Pi..Pi }
9090: function Tan(X : Float) : Float; external 'dmath';
9091: { Tangent }
9092: function ArcSin(X : Float) : Float; external 'dmath';
9093: { Arc sinus }
9094: function ArcCos(X : Float) : Float; external 'dmath';
9095: { Arc cosinus }
9096: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9097: { Angle (Ox, OM) with M(X,Y) }
9098: { -----
9099:   Hyperbolic functions
9100: ----- }
9101: function Sinh(X : Float) : Float; external 'dmath';
9102: { Hyperbolic sine }
9103: function Cosh(X : Float) : Float; external 'dmath';
9104: { Hyperbolic cosine }
9105: function Tanh(X : Float) : Float; external 'dmath';
9106: { Hyperbolic tangent }
9107: function ArcSinh(X : Float) : Float; external 'dmath';
9108: { Inverse hyperbolic sine }
9109: function ArcCosh(X : Float) : Float; external 'dmath';
9110: { Inverse hyperbolic cosine }
9111: function ArcTanh(X : Float) : Float; external 'dmath';
9112: { Inverse hyperbolic tangent }
9113: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9114: { Sinh & Cosh }
9115: { -----
9116:   Gamma function and related functions
9117: ----- }
9118: function Fact(N : Integer) : Float; external 'dmath';
9119: { Factorial }
9120: function SgnGamma(X : Float) : Integer; external 'dmath';
9121: { Sign of Gamma function }
9122: function Gamma(X : Float) : Float; external 'dmath';
9123: { Gamma function }
9124: function LnGamma(X : Float) : Float; external 'dmath';
9125: { Logarithm of Gamma function }
9126: function Stirling(X : Float) : Float; external 'dmath';
9127: { Stirling's formula for the Gamma function }
9128: function StirLog(X : Float) : Float; external 'dmath';
9129: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9130: function DiGamma(X : Float) : Float; external 'dmath';
9131: { Digamma function }
9132: function TriGamma(X : Float) : Float; external 'dmath';
9133: { Trigamma function }
9134: function IGamma(A, X : Float) : Float; external 'dmath';
9135: { Incomplete Gamma function }
9136: function JGamma(A, X : Float) : Float; external 'dmath';
9137: { Complement of incomplete Gamma function }
9138: function InvGamma(A, P : Float) : Float; external 'dmath';
9139: { Inverse of incomplete Gamma function }
9140: function Erf(X : Float) : Float; external 'dmath';
9141: { Error function }
9142: function Erfc(X : Float) : Float; external 'dmath';
9143: { Complement of error function }
9144: { -----
9145:   Beta function and related functions
9146: ----- }
9147: function Beta(X, Y : Float) : Float; external 'dmath';
9148: { Beta function }
9149: function IBeta(A, B, X : Float) : Float; external 'dmath';
9150: { Incomplete Beta function }
9151: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9152: { Inverse of incomplete Beta function }
9153: { -----
9154:   Lambert's function
9155: ----- }
9156: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9157: { -----
9158:   Binomial distribution
9159: ----- }
9160: function Binomial(N, K : Integer) : Float; external 'dmath';
9161: { Binomial coefficient C(N,K) }
9162: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9163: { Probability of binomial distribution }
9164: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';

```

```

9165: { Cumulative probability for binomial distrib. }
9166: { ----- }
9167:   Poisson distribution
9168: { ----- }
9169: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9170: { Probability of Poisson distribution }
9171: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9172: { Cumulative probability for Poisson distrib. }
9173: { ----- }
9174:   Exponential distribution
9175: { ----- }
9176: function DExpo(A, X : Float) : Float; external 'dmath';
9177: { Density of exponential distribution with parameter A }
9178: function FExpo(A, X : Float) : Float; external 'dmath';
9179: { Cumulative probability function for exponential dist. with parameter A }
9180: { ----- }
9181:   Standard normal distribution
9182: { ----- }
9183: function DNorm(X : Float) : Float; external 'dmath';
9184: { Density of standard normal distribution }
9185: function FNorm(X : Float) : Float; external 'dmath';
9186: { Cumulative probability for standard normal distrib. }
9187: function PNorm(X : Float) : Float; external 'dmath';
9188: { Prob(|U| > X) for standard normal distrib. }
9189: function InvNorm(P : Float) : Float; external 'dmath';
9190: { Inverse of standard normal distribution }
9191: { ----- }
9192:   Student's distribution
9193: { ----- }
9194: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9195: { Density of Student distribution with Nu d.o.f. }
9196: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9197: { Cumulative probability for Student distrib. with Nu d.o.f. }
9198: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9199: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9200: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9201: { Inverse of Student's t-distribution function }
9202: { ----- }
9203:   Khi-2 distribution
9204: { ----- }
9205: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9206: { Density of Khi-2 distribution with Nu d.o.f. }
9207: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9208: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9209: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9210: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9211: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9212: { Inverse of Khi-2 distribution function }
9213: { ----- }
9214:   Fisher-Snedecor distribution
9215: { ----- }
9216: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9217: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9218: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9219: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9220: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9221: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9222: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9223: { Inverse of Snedecor's F-distribution function }
9224: { ----- }
9225:   Beta distribution
9226: { ----- }
9227: function DBeta(A, B, X : Float) : Float; external 'dmath';
9228: { Density of Beta distribution with parameters A and B }
9229: function FBeta(A, B, X : Float) : Float; external 'dmath';
9230: { Cumulative probability for Beta distrib. with param. A and B }
9231: { ----- }
9232:   Gamma distribution
9233: { ----- }
9234: function DGamma(A, B, X : Float) : Float; external 'dmath';
9235: { Density of Gamma distribution with parameters A and B }
9236: function FGamma(A, B, X : Float) : Float; external 'dmath';
9237: { Cumulative probability for Gamma distrib. with param. A and B }
9238: { ----- }
9239:   Expression evaluation
9240: { ----- }
9241: function InitEval : Integer; external 'dmath';
9242: { Initializes built-in functions and returns their number }
9243: function Eval(ExpressionString : String) : Float; external 'dmath';
9244: { Evaluates an expression at run-time }
9245: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9246: { Assigns a value to a variable }
9247: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9248: { Adds a function to the parser }
9249: { ----- }
9250:   Matrices and linear equations
9251: { ----- }
9252: procedure GaussJordan(A          : TMatrix;
9253:                       Lb, Ub1, Ub2 : Integer;

```

```

9254:             var Det      : Float); external 'dmath';
9255: { Transforms a matrix according to the Gauss-Jordan method }
9256: procedure LinEq(A      : TMatrix;
9257:               B      : TVector;
9258:               Lb, Ub  : Integer;
9259:               var Det : Float); external 'dmath';
9260: { Solves a linear system according to the Gauss-Jordan method }
9261: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9262: { Cholesky factorization of a positive definite symmetric matrix }
9263: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9264: { LU decomposition }
9265: procedure LU_Solve(A      : TMatrix;
9266:                  B      : TVector;
9267:                  Lb, Ub  : Integer;
9268:                  X      : TVector); external 'dmath';
9269: { Solution of linear system from LU decomposition }
9270: procedure QR_Decomp(A      : TMatrix;
9271:                   Lb, Ub1, Ub2 : Integer;
9272:                   R      : TMatrix); external 'dmath';
9273: { QR decomposition }
9274: procedure QR_Solve(Q, R      : TMatrix;
9275:                  B      : TVector;
9276:                  Lb, Ub1, Ub2 : Integer;
9277:                  X      : TVector); external 'dmath';
9278: { Solution of linear system from QR decomposition }
9279: procedure SV_Decomp(A      : TMatrix;
9280:                   Lb, Ub1, Ub2 : Integer;
9281:                   S      : TVector;
9282:                   V      : TMatrix); external 'dmath';
9283: { Singular value decomposition }
9284: procedure SV_SetZero(S      : TVector;
9285:                   Lb, Ub : Integer;
9286:                   Tol   : Float); external 'dmath';
9287: { Set lowest singular values to zero }
9288: procedure SV_Solve(U      : TMatrix;
9289:                  S      : TVector;
9290:                  V      : TMatrix;
9291:                  B      : TVector;
9292:                  Lb, Ub1, Ub2 : Integer;
9293:                  X      : TVector); external 'dmath';
9294: { Solution of linear system from SVD }
9295: procedure SV_Approx(U      : TMatrix;
9296:                   S      : TVector;
9297:                   V      : TMatrix;
9298:                   Lb, Ub1, Ub2 : Integer;
9299:                   A      : TMatrix); external 'dmath';
9300: { Matrix approximation from SVD }
9301: procedure EigenVals(A      : TMatrix;
9302:                   Lb, Ub : Integer;
9303:                   Lambda : TCompVector); external 'dmath';
9304: { Eigenvalues of a general square matrix }
9305: procedure EigenVect(A      : TMatrix;
9306:                   Lb, Ub : Integer;
9307:                   Lambda : TCompVector;
9308:                   V      : TMatrix); external 'dmath';
9309: { Eigenvalues and eigenvectors of a general square matrix }
9310: procedure EigenSym(A      : TMatrix;
9311:                   Lb, Ub : Integer;
9312:                   Lambda : TVector;
9313:                   V      : TMatrix); external 'dmath';
9314: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9315: procedure Jacobi(A      : TMatrix;
9316:                 Lb, Ub, MaxIter : Integer;
9317:                 Tol           : Float;
9318:                 Lambda        : TVector;
9319:                 V              : TMatrix); external 'dmath';
9320: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9321: { -----
9322:   Optimization
9323: ----- }
9324: procedure MinBrack(Func      : TFunc;
9325:                   var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9326: { Brackets a minimum of a function }
9327: procedure GoldSearch(Func      : TFunc;
9328:                     A, B      : Float;
9329:                     MaxIter   : Integer;
9330:                     Tol       : Float;
9331:                     var Xmin, Ymin : Float); external 'dmath';
9332: { Minimization of a function of one variable (golden search) }
9333: procedure LinMin(Func      : TFuncNVar;
9334:                 X, DeltaX : TVector;
9335:                 Lb, Ub    : Integer;
9336:                 var R      : Float;
9337:                 MaxIter   : Integer;
9338:                 Tol       : Float;
9339:                 var F_min : Float); external 'dmath';
9340: { Minimization of a function of several variables along a line }
9341: procedure Newton(Func      : TFuncNVar;
9342:                 HessGrad  : THessGrad;

```



```

9343:         X           : TVector;
9344:         Lb, Ub       : Integer;
9345:         MaxIter      : Integer;
9346:         Tol          : Float;
9347:         var F_min     : Float;
9348:         G             : TVector;
9349:         H_inv        : TMatrix;
9350:         var Det       : Float); external 'dmath';
9351: { Minimization of a function of several variables (Newton's method) }
9352: procedure SaveNewton(FileName : string); external 'dmath';
9353: { Save Newton iterations in a file }
9354: procedure Marquardt(Func      : TFuncNVar;
9355:                    HessGrad  : THessGrad;
9356:                    X          : TVector;
9357:                    Lb, Ub     : Integer;
9358:                    MaxIter    : Integer;
9359:                    Tol        : Float;
9360:                    var F_min   : Float;
9361:                    G          : TVector;
9362:                    H_inv      : TMatrix;
9363:                    var Det     : Float); external 'dmath';
9364: { Minimization of a function of several variables (Marquardt's method) }
9365: procedure SaveMarquardt(FileName : string); external 'dmath';
9366: { Save Marquardt iterations in a file }
9367: procedure BFGS(Func      : TFuncNVar;
9368:               Gradient  : TGradient;
9369:               X          : TVector;
9370:               Lb, Ub     : Integer;
9371:               MaxIter    : Integer;
9372:               Tol        : Float;
9373:               var F_min   : Float;
9374:               G          : TVector;
9375:               H_inv      : TMatrix); external 'dmath';
9376: { Minimization of a function of several variables (BFGS method) }
9377: procedure SaveBFGS(FileName : string); external 'dmath';
9378: { Save BFGS iterations in a file }
9379: procedure Simplex(Func      : TFuncNVar;
9380:                  X          : TVector;
9381:                  Lb, Ub     : Integer;
9382:                  MaxIter    : Integer;
9383:                  Tol        : Float;
9384:                  var F_min   : Float); external 'dmath';
9385: { Minimization of a function of several variables (Simplex) }
9386: procedure SaveSimplex(FileName : string); external 'dmath';
9387: { Save Simplex iterations in a file }
9388: { -----
9389:   Nonlinear equations
9390:   ----- }
9391: procedure RootBrack(Func      : TFunc;
9392:                   var X, Y, FX, FY : Float); external 'dmath';
9393: { Brackets a root of function Func between X and Y }
9394: procedure Bisect(Func      : TFunc;
9395:                 var X, Y : Float;
9396:                 MaxIter  : Integer;
9397:                 Tol      : Float;
9398:                 var F     : Float); external 'dmath';
9399: { Bisection method }
9400: procedure Secant(Func      : TFunc;
9401:                 var X, Y : Float;
9402:                 MaxIter  : Integer;
9403:                 Tol      : Float;
9404:                 var F     : Float); external 'dmath';
9405: { Secant method }
9406: procedure NewtEq(Func, Deriv : TFunc;
9407:                 var X       : Float;
9408:                 MaxIter    : Integer;
9409:                 Tol        : Float;
9410:                 var F       : Float); external 'dmath';
9411: { Newton-Raphson method for a single nonlinear equation }
9412: procedure NewtEqs(Equations : TEquations;
9413:                 Jacobian    : TJacobian;
9414:                 X, F        : TVector;
9415:                 Lb, Ub      : Integer;
9416:                 MaxIter     : Integer;
9417:                 Tol         : Float); external 'dmath';
9418: { Newton-Raphson method for a system of nonlinear equations }
9419: procedure Broyden(Equations : TEquations;
9420:                 X, F        : TVector;
9421:                 Lb, Ub      : Integer;
9422:                 MaxIter     : Integer;
9423:                 Tol         : Float); external 'dmath';
9424: { Broyden's method for a system of nonlinear equations }
9425: { -----
9426:   Polynomials and rational fractions
9427:   ----- }
9428: function Poly(X       : Float;
9429:              Coef     : TVector;
9430:              Deg       : Integer) : Float; external 'dmath';
9431: { Evaluates a polynomial }

```

```

9432: function RFrac(X          : Float;
9433:               Coef        : TVector;
9434:               Deg1, Deg2 : Integer) : Float; external 'dmath';
9435: { Evaluates a rational fraction }
9436: function RootPol1(A, B : Float;
9437:                 var X : Float) : Integer; external 'dmath';
9438: { Solves the linear equation A + B * X = 0 }
9439: function RootPol2(Coef : TVector;
9440:                 Z      : TCompVector) : Integer; external 'dmath';
9441: { Solves a quadratic equation }
9442: function RootPol3(Coef : TVector;
9443:                 Z      : TCompVector) : Integer; external 'dmath';
9444: { Solves a cubic equation }
9445: function RootPol4(Coef : TVector;
9446:                 Z      : TCompVector) : Integer; external 'dmath';
9447: { Solves a quartic equation }
9448: function RootPol(Coef : TVector;
9449:                 Deg   : Integer;
9450:                 Z      : TCompVector) : Integer; external 'dmath';
9451: { Solves a polynomial equation }
9452: function SetRealRoots(Deg : Integer;
9453:                     Z      : TCompVector;
9454:                     Tol : Float) : Integer; external 'dmath';
9455: { Set the imaginary part of a root to zero }
9456: procedure SortRoots(Deg : Integer;
9457:                   Z      : TCompVector); external 'dmath';
9458: { Sorts the roots of a polynomial }
9459: { -----
9460: Numerical integration and differential equations
9461: ----- }
9462: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9463: { Integration by trapezoidal rule }
9464: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9465: { Integral from A to B }
9466: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9467: { Integral from 0 to B }
9468: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9469: { Convolution product at time T }
9470: procedure ConvTrap(Func1, Func2: TFunc; T, Y: TVector; N: Integer); external 'dmath';
9471: { Convolution by trapezoidal rule }
9472: procedure RKF45(F          : TDiffEqs;
9473:               Neqn         : Integer;
9474:               Y, Yp        : TVector;
9475:               var T         : Float;
9476:               Tout, RelErr, AbsErr : Float;
9477:               var Flag      : Integer); external 'dmath';
9478: { Integration of a system of differential equations }
9479: { -----
9480: Fast Fourier Transform
9481: ----- }
9482: procedure FFT(NumSamples : Integer;
9483:             InArray, OutArray : TCompVector); external 'dmath';
9484: { Fast Fourier Transform }
9485: procedure IFFT(NumSamples : Integer;
9486:             InArray, OutArray : TCompVector); external 'dmath';
9487: { Inverse Fast Fourier Transform }
9488: procedure FFT_Integer(NumSamples : Integer;
9489:                   RealIn, ImagIn : TIntVector;
9490:                   OutArray       : TCompVector); external 'dmath';
9491: { Fast Fourier Transform for integer data }
9492: procedure FFT_Integer_Cleanup; external 'dmath';
9493: { Clear memory after a call to FFT_Integer }
9494: procedure CalcFrequency(NumSamples,
9495:                       FrequencyIndex : Integer;
9496:                       InArray         : TCompVector;
9497:                       var FFT         : Complex); external 'dmath';
9498: { Direct computation of Fourier transform }
9499: { -----
9500: Random numbers
9501: ----- }
9502: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9503: { Select generator }
9504: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9505: { Initialize generator }
9506: function IRanGen : RNG_IntType; external 'dmath';
9507: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9508: function IRanGen31 : RNG_IntType; external 'dmath';
9509: { 31-bit random integer in [0 .. 2^31 - 1] }
9510: function RanGen1 : Float; external 'dmath';
9511: { 32-bit random real in [0,1] }
9512: function RanGen2 : Float; external 'dmath';
9513: { 32-bit random real in [0,1] }
9514: function RanGen3 : Float; external 'dmath';
9515: { 32-bit random real in (0,1) }
9516: function RanGen53 : Float; external 'dmath';
9517: { 53-bit random real in [0,1] }
9518: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9519: { Initializes the 'Multiply with carry' random number generator }
9520: function IRanMWC : RNG_IntType; external 'dmath';

```

```

9521: { Returns a 32 bit random number in  $[-2^{31}; 2^{31}-1]$  }
9522: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9523: { Initializes Mersenne Twister generator with a seed }
9524: procedure InitMTbyArray(InitKey : array of RNG_LongType;
9525:   KeyLength : Word); external 'dmath';
9526: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9527: function IRanMT : RNG_IntType; external 'dmath';
9528: { Random integer from MT generator }
9529: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9530: { Initializes the UVAG generator with a string }
9531: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9532: { Initializes the UVAG generator with an integer }
9533: function IRanUVAG : RNG_IntType; external 'dmath';
9534: { Random integer from UVAG generator }
9535: function RanGaussStd : Float; external 'dmath';
9536: { Random number from standard normal distribution }
9537: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9538: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9539: procedure RanMult(M : TVector; L : TMatrix;
9540:   Lb, Ub : Integer;
9541:   X : TVector); external 'dmath';
9542: { Random vector from multinormal distribution (correlated) }
9543: procedure RanMultIndep(M, S : TVector;
9544:   Lb, Ub : Integer;
9545:   X : TVector); external 'dmath';
9546: { Random vector from multinormal distribution (uncorrelated) }
9547: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9548: { Initializes Metropolis-Hastings parameters }
9549: procedure GetMHParams(var NCycles, MaxSim, SavedSim: Integer); external 'dmath';
9550: { Returns Metropolis-Hastings parameters }
9551: procedure Hastings(Func : TFuncNVar;
9552:   T : Float;
9553:   X : TVector;
9554:   V : TMatrix;
9555:   Lb, Ub : Integer;
9556:   Xmat : TMatrix;
9557:   X_min : TVector;
9558:   var F_min : Float); external 'dmath';
9559: { Simulation of a probability density function by Metropolis-Hastings }
9560: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9561: { Initializes Simulated Annealing parameters }
9562: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9563: { Initializes log file }
9564: procedure SimAnn(Func : TFuncNVar;
9565:   X, Xmin, Xmax : TVector;
9566:   Lb, Ub : Integer;
9567:   var F_min : Float); external 'dmath';
9568: { Minimization of a function of several var. by simulated annealing }
9569: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9570: { Initializes Genetic Algorithm parameters }
9571: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9572: { Initializes log file }
9573: procedure GenAlg(Func : TFuncNVar;
9574:   X, Xmin, Xmax : TVector;
9575:   Lb, Ub : Integer;
9576:   var F_min : Float); external 'dmath';
9577: { Minimization of a function of several var. by genetic algorithm }
9578: { -----
9579:   Statistics
9580: ----- }
9581: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9582: { Mean of sample X }
9583: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9584: { Minimum of sample X }
9585: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9586: { Maximum of sample X }
9587: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9588: { Median of sample X }
9589: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9590: { Standard deviation estimated from sample X }
9591: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9592: { Standard deviation of population }
9593: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9594: { Correlation coefficient }
9595: function Skewness(X : TVector; Lb, Ub : Integer; M, Sigma: Float) : Float; external 'dmath';
9596: { Skewness of sample X }
9597: function Kurtosis(X : TVector; Lb, Ub : Integer; M, Sigma: Float) : Float; external 'dmath';
9598: { Kurtosis of sample X }
9599: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9600: { Quick sort (ascending order) }
9601: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9602: { Quick sort (descending order) }
9603: procedure Interval(X1, X2 : Float;
9604:   MinDiv, MaxDiv : Integer;
9605:   var Min, Max, Step : Float); external 'dmath';
9606: { Determines an interval for a set of values }
9607: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9608:   var XMin, XMax, XStep : Float); external 'dmath';
9609: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }

```

```

9610: procedure StudIndep(N1, N2      : Integer;
9611:                    M1, M2, S1, S2 : Float;
9612:                    var T          : Float;
9613:                    var DoF        : Integer); external 'dmath';
9614: { Student t-test for independent samples }
9615: procedure StudPaired(X, Y      : TVector;
9616:                    Lb, Ub      : Integer;
9617:                    var T        : Float;
9618:                    var DoF      : Integer); external 'dmath';
9619: { Student t-test for paired samples }
9620: procedure AnOVal(Ns      : Integer;
9621:                 N        : TIntVector;
9622:                 M, S      : TVector;
9623:                 var V_f, V_r, F : Float;
9624:                 var DoF_f, DoF_r : Integer); external 'dmath';
9625: { One-way analysis of variance }
9626: procedure AnOVA2(NA, NB, Nobs : Integer;
9627:                 M, S          : TMatrix;
9628:                 V, F          : TVector;
9629:                 DoF           : TIntVector); external 'dmath';
9630: { Two-way analysis of variance }
9631: procedure Snedecor(N1, N2      : Integer;
9632:                  S1, S2        : Float;
9633:                  var F          : Float;
9634:                  var DoF1, DoF2 : Integer); external 'dmath';
9635: { Snedecor's F-test (comparison of two variances) }
9636: procedure Bartlett(Ns      : Integer;
9637:                  N        : TIntVector;
9638:                  S        : TVector;
9639:                  var Khi2   : Float;
9640:                  var DoF    : Integer); external 'dmath';
9641: { Bartlett's test (comparison of several variances) }
9642: procedure Mann_Whitney(N1, N2 : Integer;
9643:                      X1, X2   : TVector;
9644:                      var U, Eps : Float); external 'dmath';
9645: { Mann-Whitney test }
9646: procedure Wilcoxon(X, Y      : TVector;
9647:                  Lb, Ub      : Integer;
9648:                  var Ndiff    : Integer;
9649:                  var T, Eps    : Float); external 'dmath';
9650: { Wilcoxon test }
9651: procedure Kruskal_Wallis(Ns      : Integer;
9652:                         N        : TIntVector;
9653:                         X        : TMatrix;
9654:                         var H     : Float;
9655:                         var DoF   : Integer); external 'dmath';
9656: { Kruskal-Wallis test }
9657: procedure Khi2_Conform(N_cls      : Integer;
9658:                      N_estim      : Integer;
9659:                      Obs           : TIntVector;
9660:                      Calc          : TVector;
9661:                      var Khi2      : Float;
9662:                      var DoF       : Integer); external 'dmath';
9663: { Khi-2 test for conformity }
9664: procedure Khi2_Indep(N_lin      : Integer;
9665:                    N_col       : Integer;
9666:                    Obs          : TIntMatrix;
9667:                    var Khi2     : Float;
9668:                    var DoF      : Integer); external 'dmath';
9669: { Khi-2 test for independence }
9670: procedure Woolf_Conform(N_cls      : Integer;
9671:                      N_estim      : Integer;
9672:                      Obs           : TIntVector;
9673:                      Calc          : TVector;
9674:                      var G         : Float;
9675:                      var DoF       : Integer); external 'dmath';
9676: { Woolf's test for conformity }
9677: procedure Woolf_Indep(N_lin      : Integer;
9678:                    N_col       : Integer;
9679:                    Obs          : TIntMatrix;
9680:                    var G        : Float;
9681:                    var DoF       : Integer); external 'dmath';
9682: { Woolf's test for independence }
9683: procedure DimStatClassVector(var C : TStatClassVector;
9684:                             Ub      : Integer); external 'dmath';
9685: { Allocates an array of statistical classes: C[0..Ub] }
9686: procedure Distrib(X      : TVector;
9687:                 Lb, Ub    : Integer;
9688:                 A, B, H    : Float;
9689:                 C          : TStatClassVector); external 'dmath';
9690: { Distributes an array X[Lb..Ub] into statistical classes }
9691: { -----
9692:   Linear / polynomial regression
9693:   ----- }
9694: procedure LinFit(X, Y      : TVector;
9695:                 Lb, Ub      : Integer;
9696:                 B           : TVector;
9697:                 V           : TMatrix); external 'dmath';
9698: { Linear regression : Y = B(0) + B(1) * X }

```



```

9699: procedure WLinFit(X, Y, S : TVector;
9700:                   Lb, Ub : Integer;
9701:                   B      : TVector;
9702:                   V      : TMatrix); external 'dmath';
9703: { Weighted linear regression :  $Y = B(0) + B(1) * X$  }
9704: procedure SVDLinFit(X, Y : TVector;
9705:                    Lb, Ub : Integer;
9706:                    SVDtol : Float;
9707:                    B      : TVector;
9708:                    V      : TMatrix); external 'dmath';
9709: { Unweighted linear regression by singular value decomposition }
9710: procedure WSVDLinFit(X, Y, S : TVector;
9711:                    Lb, Ub : Integer;
9712:                    SVDtol : Float;
9713:                    B      : TVector;
9714:                    V      : TMatrix); external 'dmath';
9715: { Weighted linear regression by singular value decomposition }
9716: procedure MulFit(X      : TMatrix;
9717:                 Y      : TVector;
9718:                 Lb, Ub, Nvar : Integer;
9719:                 ConstTerm : Boolean;
9720:                 B          : TVector;
9721:                 V          : TMatrix); external 'dmath';
9722: { Multiple linear regression by Gauss-Jordan method }
9723: procedure WMulFit(X      : TMatrix;
9724:                 Y, S    : TVector;
9725:                 Lb, Ub, Nvar : Integer;
9726:                 ConstTerm : Boolean;
9727:                 B          : TVector;
9728:                 V          : TMatrix); external 'dmath';
9729: { Weighted multiple linear regression by Gauss-Jordan method }
9730: procedure SVDFit(X      : TMatrix;
9731:                 Y      : TVector;
9732:                 Lb, Ub, Nvar : Integer;
9733:                 ConstTerm : Boolean;
9734:                 SVDtol    : Float;
9735:                 B          : TVector;
9736:                 V          : TMatrix); external 'dmath';
9737: { Multiple linear regression by singular value decomposition }
9738: procedure WSVDFit(X      : TMatrix;
9739:                 Y, S    : TVector;
9740:                 Lb, Ub, Nvar : Integer;
9741:                 ConstTerm : Boolean;
9742:                 SVDtol    : Float;
9743:                 B          : TVector;
9744:                 V          : TMatrix); external 'dmath';
9745: { Weighted multiple linear regression by singular value decomposition }
9746: procedure PolFit(X, Y : TVector;
9747:                 Lb, Ub, Deg : Integer;
9748:                 B          : TVector;
9749:                 V          : TMatrix); external 'dmath';
9750: { Polynomial regression by Gauss-Jordan method }
9751: procedure WPolFit(X, Y, S : TVector;
9752:                 Lb, Ub, Deg : Integer;
9753:                 B          : TVector;
9754:                 V          : TMatrix); external 'dmath';
9755: { Weighted polynomial regression by Gauss-Jordan method }
9756: procedure SVDPolFit(X, Y : TVector;
9757:                   Lb, Ub, Deg : Integer;
9758:                   SVDtol    : Float;
9759:                   B          : TVector;
9760:                   V          : TMatrix); external 'dmath';
9761: { Unweighted polynomial regression by singular value decomposition }
9762: procedure WSVDPolFit(X, Y, S : TVector;
9763:                   Lb, Ub, Deg : Integer;
9764:                   SVDtol    : Float;
9765:                   B          : TVector;
9766:                   V          : TMatrix); external 'dmath';
9767: { Weighted polynomial regression by singular value decomposition }
9768: procedure RegTest(Y, Ycalc : TVector;
9769:                 LbY, UbY : Integer;
9770:                 V        : TMatrix;
9771:                 LbV, UbV : Integer;
9772:                 var Test : TRegTest); external 'dmath';
9773: { Test of unweighted regression }
9774: procedure WRegTest(Y, Ycalc, S : TVector;
9775:                 LbY, UbY : Integer;
9776:                 V        : TMatrix;
9777:                 LbV, UbV : Integer;
9778:                 var Test : TRegTest); external 'dmath';
9779: { Test of weighted regression }
9780: { -----
9781:   Nonlinear regression
9782:   ----- }
9783: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9784: { Sets the optimization algorithm for nonlinear regression }
9785: function GetOptAlgo : TOptAlgo; external 'dmath';
9786: { Returns the optimization algorithm }
9787: procedure SetMaxParam(N : Byte); external 'dmath';

```

```

9788: { Sets the maximum number of regression parameters for nonlinear regression }
9789: function GetMaxParam : Byte; external 'dmath';
9790: { Returns the maximum number of regression parameters for nonlinear regression }
9791: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9792: { Sets the bounds on the I-th regression parameter }
9793: procedure GetParamBounds(I : Byte; var ParamMin, ParamMax:Float); external 'dmath';
9794: { Returns the bounds on the I-th regression parameter }
9795: procedure NLFit(RegFunc : TRegFunc;
9796:               DerivProc : TDerivProc;
9797:               X, Y      : TVector;
9798:               Lb, Ub    : Integer;
9799:               MaxIter   : Integer;
9800:               Tol       : Float;
9801:               B         : TVector;
9802:               FirstPar,
9803:               LastPar   : Integer;
9804:               V         : TMatrix); external 'dmath';
9805: { Unweighted nonlinear regression }
9806: procedure WNLFit(RegFunc : TRegFunc;
9807:                 DerivProc : TDerivProc;
9808:                 X, Y, S    : TVector;
9809:                 Lb, Ub    : Integer;
9810:                 MaxIter   : Integer;
9811:                 Tol       : Float;
9812:                 B         : TVector;
9813:                 FirstPar,
9814:                 LastPar   : Integer;
9815:                 V         : TMatrix); external 'dmath';
9816: { Weighted nonlinear regression }
9817: procedure SetMCFile(FileName : String); external 'dmath';
9818: { Set file for saving MCMC simulations }
9819: procedure SimFit(RegFunc : TRegFunc;
9820:                 X, Y      : TVector;
9821:                 Lb, Ub    : Integer;
9822:                 B         : TVector;
9823:                 FirstPar,
9824:                 LastPar   : Integer;
9825:                 V         : TMatrix); external 'dmath';
9826: { Simulation of unweighted nonlinear regression by MCMC }
9827: procedure WSimFit(RegFunc : TRegFunc;
9828:                  X, Y, S    : TVector;
9829:                  Lb, Ub    : Integer;
9830:                  B         : TVector;
9831:                  FirstPar,
9832:                  LastPar   : Integer;
9833:                  V         : TMatrix); external 'dmath';
9834: { Simulation of weighted nonlinear regression by MCMC }
9835: { -----
9836:   Nonlinear regression models
9837:   ----- }
9838: procedure FracFit(X, Y      : TVector;
9839:                  Lb, Ub    : Integer;
9840:                  Deg1, Deg2 : Integer;
9841:                  ConsTerm   : Boolean;
9842:                  MaxIter    : Integer;
9843:                  Tol        : Float;
9844:                  B          : TVector;
9845:                  V          : TMatrix); external 'dmath';
9846: { Unweighted fit of rational fraction }
9847: procedure WFracFit(X, Y, S    : TVector;
9848:                   Lb, Ub    : Integer;
9849:                   Deg1, Deg2 : Integer;
9850:                   ConsTerm   : Boolean;
9851:                   MaxIter    : Integer;
9852:                   Tol        : Float;
9853:                   B          : TVector;
9854:                   V          : TMatrix); external 'dmath';
9855: { Weighted fit of rational fraction }
9856:
9857: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9858: { Returns the value of the rational fraction at point X }
9859: procedure ExpFit(X, Y      : TVector;
9860:                 Lb, Ub, Nexp : Integer;
9861:                 ConsTerm    : Boolean;
9862:                 MaxIter     : Integer;
9863:                 Tol         : Float;
9864:                 B           : TVector;
9865:                 V           : TMatrix); external 'dmath';
9866: { Unweighted fit of sum of exponentials }
9867: procedure WExpFit(X, Y, S    : TVector;
9868:                  Lb, Ub, Nexp : Integer;
9869:                  ConsTerm    : Boolean;
9870:                  MaxIter     : Integer;
9871:                  Tol         : Float;
9872:                  B           : TVector;
9873:                  V           : TMatrix); external 'dmath';
9874: { Weighted fit of sum of exponentials }
9875: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9876: { Returns the value of the regression function at point X }

```

```

9877: procedure IncExpFit(X, Y      : TVector;
9878:                    Lb, Ub     : Integer;
9879:                    ConstTerm  : Boolean;
9880:                    MaxIter    : Integer;
9881:                    Tol        : Float;
9882:                    B          : TVector;
9883:                    V          : TMatrix); external 'dmath';
9884: { Unweighted fit of model of increasing exponential }
9885: procedure WIncExpFit(X, Y, S   : TVector;
9886:                    Lb, Ub     : Integer;
9887:                    ConstTerm  : Boolean;
9888:                    MaxIter    : Integer;
9889:                    Tol        : Float;
9890:                    B          : TVector;
9891:                    V          : TMatrix); external 'dmath';
9892: { Weighted fit of increasing exponential }
9893: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9894: { Returns the value of the regression function at point X }
9895: procedure ExpLinFit(X, Y      : TVector;
9896:                    Lb, Ub     : Integer;
9897:                    MaxIter    : Integer;
9898:                    Tol        : Float;
9899:                    B          : TVector;
9900:                    V          : TMatrix); external 'dmath';
9901: { Unweighted fit of the "exponential + linear" model }
9902: procedure WExpLinFit(X, Y, S   : TVector;
9903:                    Lb, Ub     : Integer;
9904:                    MaxIter    : Integer;
9905:                    Tol        : Float;
9906:                    B          : TVector;
9907:                    V          : TMatrix); external 'dmath';
9908: { Weighted fit of the "exponential + linear" model }
9909:
9910: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9911: { Returns the value of the regression function at point X }
9912: procedure MichFit(X, Y      : TVector;
9913:                  Lb, Ub     : Integer;
9914:                  MaxIter    : Integer;
9915:                  Tol        : Float;
9916:                  B          : TVector;
9917:                  V          : TMatrix); external 'dmath';
9918: { Unweighted fit of Michaelis equation }
9919: procedure WMichFit(X, Y, S   : TVector;
9920:                  Lb, Ub     : Integer;
9921:                  MaxIter    : Integer;
9922:                  Tol        : Float;
9923:                  B          : TVector;
9924:                  V          : TMatrix); external 'dmath';
9925: { Weighted fit of Michaelis equation }
9926: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9927: { Returns the value of the Michaelis equation at point X }
9928: procedure MintFit(X, Y      : TVector;
9929:                  Lb, Ub     : Integer;
9930:                  MintVar    : TMintVar;
9931:                  Fit_S0     : Boolean;
9932:                  MaxIter    : Integer;
9933:                  Tol        : Float;
9934:                  B          : TVector;
9935:                  V          : TMatrix); external 'dmath';
9936: { Unweighted fit of the integrated Michaelis equation }
9937: procedure WMintFit(X, Y, S   : TVector;
9938:                  Lb, Ub     : Integer;
9939:                  MintVar    : TMintVar;
9940:                  Fit_S0     : Boolean;
9941:                  MaxIter    : Integer;
9942:                  Tol        : Float;
9943:                  B          : TVector;
9944:                  V          : TMatrix); external 'dmath';
9945: { Weighted fit of the integrated Michaelis equation }
9946: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9947: { Returns the value of the integrated Michaelis equation at point X }
9948: procedure HillFit(X, Y      : TVector;
9949:                  Lb, Ub     : Integer;
9950:                  MaxIter    : Integer;
9951:                  Tol        : Float;
9952:                  B          : TVector;
9953:                  V          : TMatrix); external 'dmath';
9954: { Unweighted fit of Hill equation }
9955: procedure WHillFit(X, Y, S   : TVector;
9956:                  Lb, Ub     : Integer;
9957:                  MaxIter    : Integer;
9958:                  Tol        : Float;
9959:                  B          : TVector;
9960:                  V          : TMatrix); external 'dmath';
9961: { Weighted fit of Hill equation }
9962: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9963: { Returns the value of the Hill equation at point X }
9964: procedure LogiFit(X, Y      : TVector;
9965:                  Lb, Ub     : Integer;

```

```

9966:          ConsTerm : Boolean;
9967:          General   : Boolean;
9968:          MaxIter   : Integer;
9969:          Tol       : Float;
9970:          B         : TVector;
9971:          V         : TMatrix); external 'dmath';
9972: { Unweighted fit of logistic function }
9973: procedure WLogiFit(X, Y, S : TVector;
9974:                  Lb, Ub : Integer;
9975:                  ConsTerm : Boolean;
9976:                  General : Boolean;
9977:                  MaxIter : Integer;
9978:                  Tol     : Float;
9979:                  B       : TVector;
9980:                  V       : TMatrix); external 'dmath';
9981: { Weighted fit of logistic function }
9982: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9983: { Returns the value of the logistic function at point X }
9984: procedure PKFit(X, Y : TVector;
9985:                Lb, Ub : Integer;
9986:                MaxIter : Integer;
9987:                Tol     : Float;
9988:                B       : TVector;
9989:                V       : TMatrix); external 'dmath';
9990: { Unweighted fit of the acid-base titration curve }
9991: procedure WPKFit(X, Y, S : TVector;
9992:                 Lb, Ub : Integer;
9993:                 MaxIter : Integer;
9994:                 Tol     : Float;
9995:                 B       : TVector;
9996:                 V       : TMatrix); external 'dmath';
9997: { Weighted fit of the acid-base titration curve }
9998: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9999: { Returns the value of the acid-base titration function at point X }
10000: procedure PowFit(X, Y : TVector;
10001:                 Lb, Ub : Integer;
10002:                 MaxIter : Integer;
10003:                 Tol     : Float;
10004:                 B       : TVector;
10005:                 V       : TMatrix); external 'dmath';
10006: { Unweighted fit of power function }
10007: procedure WPowFit(X, Y, S : TVector;
10008:                  Lb, Ub : Integer;
10009:                  MaxIter : Integer;
10010:                  Tol     : Float;
10011:                  B       : TVector;
10012:                  V       : TMatrix); external 'dmath';
10013: { Weighted fit of power function }
10014:
10015: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10016: { Returns the value of the power function at point X }
10017: procedure GammaFit(X, Y : TVector;
10018:                   Lb, Ub : Integer;
10019:                   MaxIter : Integer;
10020:                   Tol     : Float;
10021:                   B       : TVector;
10022:                   V       : TMatrix); external 'dmath';
10023: { Unweighted fit of gamma distribution function }
10024: procedure WGammaFit(X, Y, S : TVector;
10025:                    Lb, Ub : Integer;
10026:                    MaxIter : Integer;
10027:                    Tol     : Float;
10028:                    B       : TVector;
10029:                    V       : TMatrix); external 'dmath';
10030: { Weighted fit of gamma distribution function }
10031: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10032: { Returns the value of the gamma distribution function at point X }
10033: { -----
10034:   Principal component analysis
10035:   ----- }
10036: procedure VecMean(X : TMatrix;
10037:                  Lb, Ub, Nvar : Integer;
10038:                  M : TVector); external 'dmath';
10039: { Computes the mean vector M from matrix X }
10040: procedure VecSD(X : TMatrix;
10041:                Lb, Ub, Nvar : Integer;
10042:                M, S : TVector); external 'dmath';
10043: { Computes the vector of standard deviations S from matrix X }
10044: procedure MatVarCov(X : TMatrix;
10045:                    Lb, Ub, Nvar : Integer;
10046:                    M : TVector;
10047:                    V : TMatrix); external 'dmath';
10048: { Computes the variance-covariance matrix V from matrix X }
10049: procedure MatCorrel(V : TMatrix;
10050:                   Nvar : Integer;
10051:                   R : TMatrix); external 'dmath';
10052: { Computes the correlation matrix R from the var-cov matrix V }
10053: procedure PCA(R : TMatrix;
10054:              Nvar : Integer;

```



```

10055:         Lambda : TVector;
10056:         C, Rc : TMatrix); external 'dmath';
10057: { Performs a principal component analysis of the correlation matrix R }
10058: procedure ScaleVar(X : TMatrix;
10059:         Lb, Ub, Nvar : Integer;
10060:         M, S : TVector;
10061:         Z : TMatrix); external 'dmath';
10062: { Scales a set of variables by subtracting means and dividing by SD's }
10063: procedure PrinFac(Z : TMatrix;
10064:         Lb, Ub, Nvar : Integer;
10065:         C, F : TMatrix); external 'dmath';
10066: { Computes principal factors }
10067: { -----
10068:   Strings
10069:   ----- }
10070: function LTrim(S : String) : String; external 'dmath';
10071: { Removes leading blanks }
10072: function RTrim(S : String) : String; external 'dmath';
10073: { Removes trailing blanks }
10074: function Trim(S : String) : String; external 'dmath';
10075: { Removes leading and trailing blanks }
10076: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10077: { Returns a string made of character C repeated N times }
10078: function RFill(S : String; L : Byte) : String; external 'dmath';
10079: { Completes string S with trailing blanks for a total length L }
10080: function LFill(S : String; L : Byte) : String; external 'dmath';
10081: { Completes string S with leading blanks for a total length L }
10082: function CFill(S : String; L : Byte) : String; external 'dmath';
10083: { Centers string S on a total length L }
10084: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10085: { Replaces in string S all the occurrences of C1 by C2 }
10086: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10087: { Extracts a field from a string }
10088: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10089: { Parses a string into its constitutive fields }
10090: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10091: { Sets the numeric format }
10092: function FloatStr(X : Float) : String; external 'dmath';
10093: { Converts a real to a string according to the numeric format }
10094: function IntStr(N : LongInt) : String; external 'dmath';
10095: { Converts an integer to a string }
10096: function CompStr(Z : Complex) : String; external 'dmath';
10097: { Converts a complex number to a string }
10098: {$IFDEF DELPHI}
10099: function StrDec(S : String) : String; external 'dmath';
10100: { Set decimal separator to the symbol defined in SysUtils }
10101: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10102: { Test if a string represents a number and returns it in X }
10103: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10104: { Reads a floating point number from an Edit control }
10105: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10106: { Writes a floating point number in a text file }
10107: {$ENDIF}
10108: { -----
10109:   BGI / Delphi graphics
10110:   ----- }
10111: function InitGraphics
10112: {$IFDEF DELPHI}
10113: (Width, Height : Integer) : Boolean;
10114: {$ELSE}
10115: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10116: { Enters graphic mode }
10117: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10118:         X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10119: { Sets the graphic window }
10120: procedure SetOxScale(Scale : TScale;
10121:         OxMin, OxMax, OxStep : Float); external 'dmath';
10122: { Sets the scale on the Ox axis }
10123: procedure SetOyScale(Scale : TScale;
10124:         OyMin, OyMax, OyStep : Float); external 'dmath';
10125: { Sets the scale on the Oy axis }
10126: procedure GetOxScale(var Scale : TScale;
10127:         var OxMin, OxMax, OxStep : Float); external 'dmath';
10128: { Returns the scale on the Ox axis }
10129: procedure GetOyScale(var Scale : TScale;
10130:         var OyMin, OyMax, OyStep : Float); external 'dmath';
10131: { Returns the scale on the Oy axis }
10132: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10133: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10134: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10135: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10136: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10137: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10138: {$IFDEF DELPHI}
10139: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10140: { Sets the font for the main graph title }
10141: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10142: { Sets the font for the Ox axis (title and labels) }
10143: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';

```

```

10144: { Sets the font for the Oy axis (title and labels) }
10145: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10146: { Sets the font for the legends }
10147: procedure SetClipping(Clip : Boolean); external 'dmath';
10148: { Determines whether drawings are clipped at the current viewport
10149:   boundaries, according to the value of the Boolean parameter Clip }
10150: { $ENDIF }
10151: procedure PlotOxAxis({$IFDEF DELPHI}Canvas : TCanvas){$ENDIF}; external 'dmath';
10152: { Plots the horizontal axis }
10153: procedure PlotOyAxis({$IFDEF DELPHI}Canvas : TCanvas){$ENDIF}; external 'dmath';
10154: { Plots the vertical axis }
10155: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10156: { Plots a grid on the graph }
10157: procedure WriteGraphTitle({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10158: { Writes the title of the graph }
10159: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10160: { Sets the maximum number of curves and re-initializes their parameters }
10161: procedure SetPointParam
10162:   {$IFDEF DELPHI}
10163:   (CurvIndex, Symbol, Size : Integer; Color : TColor);
10164:   {$ELSE}
10165:   (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10166: { Sets the point parameters for curve # CurvIndex }
10167: procedure SetLineParam
10168:   {$IFDEF DELPHI}
10169:   (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10170:   {$ELSE}
10171:   (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10172: { Sets the line parameters for curve # CurvIndex }
10173: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10174: { Sets the legend for curve # CurvIndex }
10175: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10176: { Sets the step for curve # CurvIndex }
10177: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10178: procedure GetPointParam
10179:   {$IFDEF DELPHI}
10180:   (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10181:   {$ELSE}
10182:   (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10183: { Returns the point parameters for curve # CurvIndex }
10184: procedure GetLineParam
10185:   {$IFDEF DELPHI}
10186:   (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10187:   {$ELSE}
10188:   (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10189: { Returns the line parameters for curve # CurvIndex }
10190: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10191: { Returns the legend for curve # CurvIndex }
10192: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10193: { Returns the step for curve # CurvIndex }
10194: {$IFDEF DELPHI}
10195: procedure PlotPoint(Canvas : TCanvas;
10196:   X, Y : Float; CurvIndex : Integer); external 'dmath';
10197: {$ELSE}
10198: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10199: {$ENDIF}
10200: { Plots a point on the screen }
10201: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10202:   X, Y : TVector;
10203:   Lb, Ub, CurvIndex : Integer); external 'dmath';
10204: { Plots a curve }
10205: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10206:   X, Y, S : TVector;
10207:   Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10208: { Plots a curve with error bars }
10209: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10210:   Func : TFunc;
10211:   Xmin, Xmax : Float;
10212:   {$IFDEF DELPHI}Npt : Integer;{$ENDIF}
10213:   CurvIndex : Integer); external 'dmath';
10214: { Plots a function }
10215: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10216:   NCurv : Integer;
10217:   ShowPoints, ShowLines : Boolean); external 'dmath';
10218: { Writes the legends for the plotted curves }
10219: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10220:   Nx, Ny, Nc : Integer;
10221:   X, Y, Z : TVector;
10222:   F : TMatrix); external 'dmath';
10223: { Contour plot }
10224: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10225: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10226: function Xuser(X : Integer):Float; external 'dmath'; {Converts screen coordinate X to user abscissa }
10227: function Yuser(Y : Integer):Float; external 'dmath'; {Converts screen coordinate Y to user ordinate }
10228: {$IFDEF DELPHI}
10229: procedure LeaveGraphics; external 'dmath';
10230: { Quits graphic mode }
10231: {$ENDIF}
10232: { -----

```

```

10233:   LaTeX graphics
10234:   ----- }
10235: function TeX_InitGraphics(FileName      : String; PgWidth, PgHeight : Integer;
10236:                           Header       : Boolean) : Boolean; external 'dmath';
10237: { Initializes the LaTeX file }
10238: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10239: { Sets the graphic window }
10240: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10241: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10242: { Sets the scale on the Ox axis }
10243: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10244: { Sets the scale on the Oy axis }
10245: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10246: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10247: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10248: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10249: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10250: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10251: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10252: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10253: { Sets the maximum number of curves and re-initializes their parameters }
10254: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10255: { Sets the point parameters for curve # CurvIndex }
10256: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10257:                           Width : Float; Smooth : Boolean); external 'dmath';
10258: { Sets the line parameters for curve # CurvIndex }
10259: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10260: { Sets the legend for curve # CurvIndex }
10261: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10262: { Sets the step for curve # CurvIndex }
10263: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10264: { Plots a curve }
10265: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10266:                                     Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10267: { Plots a curve with error bars }
10268: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10269:                       Npt : Integer; CurvIndex : Integer); external 'dmath';
10270: { Plots a function }
10271: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10272: { Writes the legends for the plotted curves }
10273: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix); external 'dmath';
10274: { Contour plot }
10275: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10276: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10277:
10278: //*****unit uPSI_SynPdf;
10279: function RawUTF8ToPDFString( const Value : RawUTF8 ) : PDFString
10280: function _DateTimeToPdfDate( ADate : TDateTime ) : TPdfDate
10281: function _PdfDateToDateTime( const AText : TPdfDate ) : TDateTime
10282: function PdfRect( Left, Top, Right, Bottom : Single ) : TPdfRect;
10283: function PdfRect1( const Box : TPdfBox ) : TPdfRect;
10284: function PdfBox( Left, Top, Width, Height : Single ) : TPdfBox
10285: //Function _GetCharCount( Text : PAnsiChar ) : integer
10286: //Procedure L2R( W : PWideChar; L : integer)
10287: function PdfCoord( MM : single ) : integer
10288: function CurrentPrinterPaperSize : TPDPaperSize
10289: function CurrentPrinterRes : TPoint
10290: procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8 )
10291: procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer )
10292: procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect )
10293: Const('Uspl0','String 'uspl0.dll
10294: AddTypes('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10295:   fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10296: AddTypes('TScriptState_set', 'set of TScriptState_enum
10297: //*****
10298:
10299: procedure SIRegister_PMrand(CL: TPSPascalCompiler); //ParkMiller
10300: begin
10301:   Procedure PMrandomize( I : word)
10302:   Function PMrandom : longint
10303:   Function Rrand : extended
10304:   Function Irand( N : word) : word
10305:   Function Brand( P : extended) : boolean
10306:   Function Nrand : extended
10307: end;
10308:
10309: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10310: begin
10311:   Function Endian( x : LongWord) : LongWord
10312:   Function Endian64( x : Int64) : Int64
10313:   Function spRol( x : LongWord; y : Byte) : LongWord
10314:   Function spRor( x : LongWord; y : Byte) : LongWord
10315:   Function Ror64( x : Int64; y : Byte) : Int64
10316: end;
10317:
10318: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10319: begin
10320:   Procedure ClearModules
10321:   Procedure ReadMapFile( Fname : string)

```

```

10322: Function AddressInfo( Address : dword ) : string
10323: end;
10324:
10325: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10326: begin
10327:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10328:   + 'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10329:   + 'teByOther, tpExecuteByOther )
10330:   TTarPermissions', 'set of TTarPermission
10331:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10332:   + 'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10333:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10334:   TTarModes', 'set of TTarMode
10335:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10336:   + 'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10337:   + 'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10338:   + 'ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10339:   + 'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10340:   SIRegister_TTarArchive(CL);
10341:   SIRegister_TTarWriter(CL);
10342: Function PermissionString( Permissions : TTarPermissions ) : STRING
10343: Function ConvertFilename( Filename : STRING) : STRING
10344: Function FileTimeGMT( FileName : STRING) : TDateTime;
10345: Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10346: Procedure ClearDirRec( var DirRec : TTarDirRec)
10347: end;
10348:
10349:
10350: //*****unit uPSI_TlHelp32;
10351: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10352: begin
10353:   Const ('MAX_MODULE_NAME32', 'LongInt'( 255);
10354:   Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10355:   Const ('TH32CS_SNAPHEAPLIST', 'LongWord( $00000001);
10356:   Const ('TH32CS_SNAPPROCESS', 'LongWord').SetUInt( $00000002);
10357:   Const ('TH32CS_SNAPTHREAD', 'LongWord').SetUInt( $00000004);
10358:   Const ('TH32CS_SNAPMODULE', 'LongWord').SetUInt( $00000008);
10359:   Const ('TH32CS_INHERIT', 'LongWord').SetUInt( $80000000);
10360:   tagHEAPLIST32', 'record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10361:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10362:   AddTypeS('TheapList32', 'tagHEAPLIST32
10363:   Const ('HF32_DEFAULT', 'LongInt'( 1);
10364:   Const ('HF32_SHARED', 'LongInt'( 2);
10365:   Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10366:   Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10367:   AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10368:   + 'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10369:   + 'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10370:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10371:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10372:   Const ('LF32_FIXED', 'LongWord').SetUInt( $00000001);
10373:   Const ('LF32_FREE', 'LongWord').SetUInt( $00000002);
10374:   Const ('LF32_MOVEABLE', 'LongWord').SetUInt( $00000004);
10375:   Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10376:   Function Heap32Next( var lphe : THeapEntry32) : BOOL
10377:   DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10378:   AddTypeS('tagTHREADEENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10379:   + '2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10380:   + 'aPri : Longint; dwFlags : DWORD; end
10381:   AddTypeS('THREADEENTRY32', 'tagTHREADEENTRY32
10382:   AddTypeS('TThreadEntry32', 'tagTHREADEENTRY32
10383:   Function Thread32First( hSnapshot : THandle; var lphe : TThreadEntry32) : BOOL
10384:   Function Thread32Next( hSnapshot : THandle; var lphe : TThreadEntry32) : BOOL
10385: end;
10386:   Const ('EW_RESTARTWINDOWS', 'LongWord').SetUInt( $0042);
10387:   Const ('EW_REBOOTSYSTEM', 'LongWord( $0043);
10388:   Const ('EW_EXITANDEXECAPP', 'LongWord( $0044);
10389:   Const ('ENDSESSION_LOGOFF', 'LongWord').SetUInt( DWORD ( $80000000 ));
10390:   Const ('EWX_LOGOFF', 'LongInt'( 0);
10391:   Const ('EWX_SHUTDOWN', 'LongInt'( 1);
10392:   Const ('EWX_REBOOT', 'LongInt'( 2);
10393:   Const ('EWX_FORCE', 'LongInt'( 4);
10394:   Const ('EWX_FORCEOFF', 'LongInt'( 8);
10395:   Const ('EWX_FORCEIFHUNG', 'LongWord').SetUInt( $10);
10396:   Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10397:   Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10398:   Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10399:   Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10400:   Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10401:   Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10402:   Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10403:   Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10404:   Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10405:   Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10406:   Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10407:   Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10408:   Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10409:   Function GetDesktopWindow : HWND
10410:   Function GetParent( hWnd : HWND) : HWND

```



```

10411: Function SetParent( hWndChild, hWndNewParent : HWND ) : HWND
10412: Function GetTopWindow( hWnd : HWND ) : HWND
10413: Function GetNextWindow( hWnd : HWND; uCmd : UINT ) : HWND
10414: Function GetWindow( hWnd : HWND; uCmd : UINT ) : HWND
10415: //Delphi DFM
10416: Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10417: Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string ) : integer
10418: procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10419: function GetHighlightersFilter(AHighlighters: TStringList): string;
10420: function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10421: Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL ) : BOOL
10422: Function OpenIcon( hWnd : HWND ) : BOOL
10423: Function CloseWindow( hWnd : HWND ) : BOOL
10424: Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL ) : BOOL
10425: Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT ) : BOOL
10426: Function IsWindowVisible( hWnd : HWND ) : BOOL
10427: Function IsIconic( hWnd : HWND ) : BOOL
10428: Function AnyPopup : BOOL
10429: Function BringWindowToTop( hWnd : HWND ) : BOOL
10430: Function IsZoomed( hWnd : HWND ) : BOOL
10431: Function IsWindow( hWnd : HWND ) : BOOL
10432: Function IsMenu( hMenu : HMENU ) : BOOL
10433: Function IsChild( hWndParent, hWnd : HWND ) : BOOL
10434: Function DestroyWindow( hWnd : HWND ) : BOOL
10435: Function ShowWindow( hWnd : HWND; nCmdShow : Integer ) : BOOL
10436: Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD ) : BOOL
10437: Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer ) : BOOL
10438: Function FlashWindow( hWnd : HWND; bInvert : BOOL ) : BOOL
10439: Function IsWindowUnicode( hWnd : HWND ) : BOOL
10440: Function EnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL
10441: Function IsWindowEnabled( hWnd : HWND ) : BOOL
10442:
10443: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10444: begin
10445:   const ('ShowSetupDialogOptLong','String '--setup
10446:   PrimaryConfPathOptLong','String '--primary-config-path=
10447:   PrimaryConfPathOptShort','String '--pcp=
10448:   SecondaryConfPathOptLong','String '--secondary-config-path=
10449:   SecondaryConfPathOptShort','String '--scp=
10450:   NoSplashScreenOptLong','String '--no-splash-screen
10451:   NoSplashScreenOptShort','String '--nsc
10452:   StartedByStartLazarusOpt','String '--started-by-startlazarus
10453:   SkipLastProjectOpt','String '--skip-last-project
10454:   DebugLogOpt','String '--debug-log=
10455:   DebugLogOptEnable','String '--debug-enable=
10456:   LanguageOpt','String '--language=
10457:   LazarusDirOpt','String '--lazarusdir=
10458:   Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10459:   Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10460:   Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings ) : string
10461:   Function IsHelpRequested : Boolean
10462:   Function IsVersionRequested : boolean
10463:   Function GetLanguageSpecified : string
10464:   Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10465:   Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10466:   Procedure ParseNoGuiCmdLineParams
10467:   Function ExtractCmdLineFileNames : TStrings
10468: end;
10469:
10470:
10471: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10472: begin
10473:   Function CompareFileNames( const Filename1, Filename2 : string) : integer
10474:   Function CompareFileNamesIgnoreCase( const Filename1, Filename2 : string) : integer
10475:   Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10476:   Function CompareFileExtL( const Filename, Ext : string) : integer;
10477:   Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10478:   Function CompareFileNames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10479:   Function CompareFileNamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10480:   Function DirPathExists( DirectoryName : string) : boolean
10481:   Function DirectoryIsWritable( const DirectoryName : string) : boolean
10482:   Function ExtractFileNameOnly( const AFilename : string) : string
10483:   Function FilenameIsAbsolute( const TheFilename : string) : boolean
10484:   Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10485:   Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10486:   Function ForceDirectory( DirectoryName : string) : boolean
10487:   Procedure CheckIfFileIsExecutable( const AFilename : string)
10488:   Procedure CheckIfFileIsSymlink( const AFilename : string)
10489:   Function FileIsText( const AFilename : string) : boolean
10490:   Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10491:   Function FilenameIsTrimmed( const TheFilename : string) : boolean
10492:   Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10493:   Function TrimFilename( const AFilename : string) : string
10494:   Function ResolveDots( const AFilename : string) : string
10495:   Procedure ForcePathDelims( var FileName : string)
10496:   Function GetForcedPathDelims( const FileName : string) : string
10497:   Function CleanAndExpandFilename( const FileName : string) : string
10498:   Function CleanAndExpandDirectory( const FileName : string) : string
10499:   Function TrimAndExpandFilename( const FileName : string; const BaseDir : string) : string

```

```

10500: Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10501: Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10502: Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
AlwaysRequireSharedBaseFolder: Boolean) : string
10503: Function FileIsInPath( const Filename, Path : string) : boolean
10504: Function AppendPathDelim( const Path : string) : string
10505: Function ChompPathDelim( const Path : string) : string
10506: Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10507: Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10508: Function MinimizeSearchPath( const SearchPath : string) : string
10509: Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10510: (*Function FileExistsUTF8( const Filename : string) : boolean
10511: Function FileAgeUTF8( const FileName : string) : Longint
10512: Function DirectoryExistsUTF8( const Directory : string) : Boolean
10513: Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10514: Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10515: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10516: Procedure FindCloseUTF8( var F : TSearchRec)
10517: Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10518: Function FileGetAttrUTF8( const FileName : String) : Longint
10519: Function FileSetAttrUTF8( const FileName : String; Attr : longint) : Longint
10520: Function DeleteFileUTF8( const FileName : String) : Boolean
10521: Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10522: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10523: Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10524: Function GetCurrentDirUTF8 : String
10525: Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10526: Function CreateDirUTF8( const NewDir : String) : Boolean
10527: Function RemoveDirUTF8( const Dir : String) : Boolean
10528: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10529: Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10530: Function FileCreateUTF8( const FileName : string) : THandle;
10531: Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10532: Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10533: Function FileSizeUtf8( const FileName : string) : int64
10534: Function GetFileDescription( const AFileName : string) : string
10535: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10536: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10537: Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10538: Function IsUNCPath( const Path : String) : Boolean
10539: Function ExtractUNCVolume( const Path : String) : String
10540: Function ExtractFileRoot( FileName : String) : String
10541: Function GetDarwinSystemFilename( Filename : string) : string
10542: Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10543: Function StrToCmdLineParam( const Param : string) : string
10544: Function MergeCmdLineParams( ParamList : TStrings) : string
10545: Procedure InvalidateFileStateCache( const FileName : string)
10546: Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10547: Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10548: Function ReadFileToString( const Filename : string) : string
10549: type
10550:   TCopyFileFlag = ( cffOverwriteFile,
10551:     cffCreateDestDirectory, cffPreserveTime );
10552:   TCopyFileFlags = set of TCopyFileFlag;*)
10553:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10554:   TCopyFileFlags', 'set of TCopyFileFlag
10555: Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10556: end;
10557:
10558: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10559: begin
10560:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10561:   SIRegister_TMask(CL);
10562:   SIRegister_TParseStringList(CL);
10563:   SIRegister_TMaskList(CL);
10564: Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10565: Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10566: Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10567: Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10568: end;
10569:
10570: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10571: begin
10572:   //PShellHookInfo', '^TShellHookInfo // will not work
10573:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10574:   SHELLHOOKINFO', 'TShellHookInfo
10575:   LPSHELLHOOKINFO', 'PShellHookInfo
10576:   TJvmShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10577:   SIRegister_TJvmShellHook(CL);
10578: Function InitJvmShellHooks : Boolean
10579: Procedure UnInitJvmShellHooks
10580: end;
10581:
10582: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10583: begin
10584:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10585:     +', dcHasSetSel, dcWantTab, dcNative )
10586:   TDlgCodes', 'set of TDlgCode

```

```

10587: 'dcWantMessage', 'dcWantAllKeys);
10588: SIRegister_IJvExControl(CL);
10589: SIRegister_IJvDenySubClassing(CL);
10590: SIRegister_TStructPtrMessage(CL);
10591: Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor);
10592: Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10593: Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10594: Procedure HandleDotNetHighlighting( Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10595: Function CreateWMMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10596: Function CreateWMMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10597: Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10598: Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10599: Function GetFocusedControl( AControl : TControl) : TWinControl
10600: Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10601: Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10602: Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10603: Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10604: SIRegister_TJvExControl(CL);
10605: SIRegister_TJvExWinControl(CL);
10606: SIRegister_TJvExCustomControl(CL);
10607: SIRegister_TJvExGraphicControl(CL);
10608: SIRegister_TJvExHintWindow(CL);
10609: SIRegister_TJvExPubGraphicControl(CL);
10610: end;
10611:
10612: (*-----*)
10613: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10614: begin
10615:   Procedure EncodeStream( Input, Output : TStream)
10616:   Procedure DecodeStream( Input, Output : TStream)
10617:   Function EncodeString1( const Input : string) : string
10618:   Function DecodeString1( const Input : string) : string
10619: end;
10620:
10621: (*-----*)
10622: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10623: begin
10624:   SIRegister_TWebAppRegInfo(CL);
10625:   SIRegister_TWebAppRegList(CL);
10626:   Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10627:   Procedure RegisterWebApp( const AFileName, AProgID : string)
10628:   Procedure UnregisterWebApp( const AProgID : string)
10629:   Function FindRegisteredWebApp( const AProgID : string) : string
10630:   Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10631: 'sUDPPort','String 'UDPPort
10632: end;
10633:
10634: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10635: begin
10636:   // TStringDynArray, 'array of string
10637:   Function GetEnvVarValue( const VarName : string) : string
10638:   Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10639:   Function DeleteEnvVar( const VarName : string) : Integer
10640:   Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
BufSize:Int):Int;
10641:   Function ExpandEnvVars( const Str : string) : string
10642:   Function GetAllEnvVars( const Vars : TStrings) : Integer
10643:   Procedure GetAllEnvVarNames( const Names : TStrings);
10644:   Function GetAllEnvVarNames1 : TStringDynArray;
10645:   Function EnvBlockSize : Integer
10646:   TPJEnvVarsEnum, 'Procedure ( const VarName : string; Data : TObject)
10647:   SIRegister_TPJEnvVarsEnumerator(CL);
10648:   SIRegister_TPJEnvVars(CL);
10649:   FindClass('TOBJECT'),'EPJEnvVars
10650:   FindClass('TOBJECT'),'EPJEnvVars
10651:   //Procedure Register
10652: end;
10653:
10654: (*-----*)
10655: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10656: begin
10657:   'cOneSecInMS','LongInt'( 1000);
10658:   //'cDefTimeSlice','LongInt'( 50);
10659:   //'cDefMaxExecTime',' cOneMinInMS);
10660:   'cAppErrorMask','LongInt'( 1 shl 29);
10661:   Function IsApplicationError( const ErrCode : LongWord) : Boolean
10662:   TPJConsoleAppPriority, '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10663:   TPJConsoleColors, 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10664:   Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10665:   Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10666:   Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10667:   Function MakeSize( const ACX, ACY : LongInt) : TSize
10668:   SIRegister_TPJCustomConsoleApp(CL);
10669:   SIRegister_TPJConsoleApp(CL);
10670: end;
10671:
10672: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10673: begin
10674:   INVALID_IP_ADDRESS, 'LongWord').SetUInt( $ffffff);

```

```

10675:   t_encoding', '( uencode, base64, mime )
10676: Function internet_date( date : TDateTime) : string
10677: Function lookup_hostname( const hostname : string) : longint
10678: Function my_hostname : string
10679: Function my_ip_address : longint
10680: Function ip2string( ip_address : longint) : string
10681: Function resolve_hostname( ip : longint) : string
10682: Function address_from( const s : string; count : integer) : string
10683: Function encode_base64( data : TStream) : TStringList
10684: Function decode_base64( source : TStringList) : TMemoryStream
10685: Function posn( const s, t : string; count : integer) : integer
10686: Function poscn( c : char; const s : string; n : integer) : integer
10687: Function filename_of( const s : string) : string
10688: //Function trim( const s : string) : string
10689: //Procedure setlength( var s : string; l : byte)
10690: Function TimeZoneBias : longint
10691: Function eight2seven_quoteprint( const s : string) : string
10692: Function eight2seven_german( const s : string) : string
10693: Function seven2eight_quoteprint( const s : string) : string end;
10694: type in_addr, 'record s_bytes : array[1..4] of byte; end;
10695: Function socketerror : cint
10696: Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10697: Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10698: Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10699: //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10700: Function fplistn( s : cint; backlog : cint) : cint
10701: //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10702: //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10703: //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10704: Function NetAddrToStr( Entry : in_addr) : String
10705: Function HostAddrToStr( Entry : in_addr) : String
10706: Function StrToHostAddr( IP : String) : in_addr
10707: Function StrToNetAddr( IP : String) : in_addr
10708: SOL_SOCKET, 'LongWord').SetUInt( $ffff);
10709: cint8, 'shortint
10710: cuint8, 'byte
10711: cchar, 'cint8
10712: cschar, 'cint8
10713: cuchar, 'cuint8
10714: cint16, 'smallint
10715: cuint16, 'word
10716: cshort, 'cint16
10717: cushort, 'cint16
10718: cushort, 'cuint16
10719: cint32, 'longint
10720: cuint32, 'longword
10721: cint, 'cint32
10722: csint, 'cint32
10723: cuint, 'cuint32
10724: csigned, 'cint
10725: cunsigned, 'cuint
10726: cint64, 'int64
10727: clonglong, 'cint64
10728: cslonglong, 'cint64
10729: cbool, 'longbool
10730: cfloat, 'single
10731: cdouble, 'double
10732: clongdouble, 'extended
10733:
10734: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10735: begin
10736:   TlkJSONtypes', '( jsBase, jsNumber, jsString, jsBoolean, jsNull, jsList, jsObject )
10737:   SIRegister_TlkJSONdotnetclass(CL);
10738:   SIRegister_TlkJSONbase(CL);
10739:   SIRegister_TlkJSONnumber(CL);
10740:   SIRegister_TlkJSONstring(CL);
10741:   SIRegister_TlkJSONboolean(CL);
10742:   SIRegister_TlkJSONnull(CL);
10743:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10744:   + 'se; data : TObjct; var Continue : Boolean)
10745:   SIRegister_TlkJSONcustomlist(CL);
10746:   SIRegister_TlkJSONlist(CL);
10747:   SIRegister_TlkJSONobjectmethod(CL);
10748:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10749:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10750:   SIRegister_TlkHashTable(CL);
10751:   SIRegister_TlkBalTree(CL);
10752:   SIRegister_TlkJSONobject(CL);
10753:   SIRegister_TlkJSON(CL);
10754:   SIRegister_TlkJSONstreamed(CL);
10755: Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer) : string
10756: end;
10757:
10758: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10759: begin
10760:   TZListSortCompare', 'Function (Item1, Item2 : TObjct) : Integer
10761:   SIRegister_TZSortedList(CL);
10762: Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10763: Function zLastDelimiter( const Delimiters, Str : string) : Integer

```



```

10764: //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10765: //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10766: Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10767: Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10768: Function EndsWith( const Str, SubStr : WideString) : Boolean;
10769: Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10770: Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10771: Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10772: Function SQLStrToFloat( const Str : AnsiString) : Extended
10773: //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10774: //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10775: Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10776: Function StrToBoolEx( Str : string) : Boolean
10777: Function BoolToStrEx( Bool : Boolean) : String
10778: Function IsIpAddr( const Str : string) : Boolean
10779: Function zSplitString( const Str, Delimiters : string) : TStrings
10780: Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10781: Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10782: Function ComposeString( List : TStrings; const Delimiter : string) : string
10783: Function FloatToSQLStr( Value : Extended) : string
10784: Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10785: Function SplitStringEx( const Str, Delimiter : string) : TStrings
10786: Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10787: Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10788: Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10789: Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10790: Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10791: Function StrToBytes3( const Value : WideString) : TByteDynArray;
10792: Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10793: Function BytesToVar( const Value : TByteDynArray) : Variant
10794: Function VarToBytes( const Value : Variant) : TByteDynArray
10795: Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10796: Function TimestampStrToDateTime( const Value : string) : TDateTime
10797: Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10798: Function EncodeCString( const Value : string) : string
10799: Function DecodeCString( const Value : string) : string
10800: Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10801: Function MemPas( Buffer : PChar; Length : LongInt) : string
10802: Procedure DecodeSQLVersioning(const FullVersion:Integer;out MajorVersion:Integer;out MinorVersion:Integer;out
SubVersion:Integer);
10803: Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
SubVersion:Integer):Integer;
10804: Function FormatSQLVersion( const SQLVersion : Integer) : String
10805: Function ZStrToFloat( Value : AnsiChar) : Extended;
10806: Function ZStrToFloat1( Value : AnsiString) : Extended;
10807: Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10808: Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10809: Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10810: Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10811: Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10812: Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10813: Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10814: end;
10815:
10816: unit uPSI_ZEncoding;
10817: Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10818: Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10819: Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10820: Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10821: Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10822: Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10823: Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10824: Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10825: Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10826: Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10827: Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10828: Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10829: Function ZConvertStringToRawWithAutoEncode( const Src:String;const StringCP,RawCP:Word):RawByteString;
10830: Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10831: Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10832: Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP : Word) : UTF8String
10833: Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10834: Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP : Word) : AnsiString
10835: Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10836: Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10837: Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10838: Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10839: Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10840: Function ZConvertStringToUnicodeWithAutoEncode( const Src : String; const StringCP:Word):WideString
10841: Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10842: Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10843: Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10844: Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10845: Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10846: Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10847: Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10848: Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10849: Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10850: Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString

```

```

10851: Function ZMoveUTF8ToString(const Src : UTF8String; StringCP : Word) : String
10852: Function ZMoveStringToUTF8(const Src : String; const StringCP : Word) : UTF8String
10853: Function ZUnknownRawToUnicode(const S : RawByteString; const CP : Word) : WideString
10854: Function ZUnknownRawToUnicodeWithAutoEncode(const S : RawByteString; const CP : Word) : WideString
10855: Function ZUnicodeToUnknownRaw(const US : WideString; CP : Word) : RawByteString
10856: Function ZDefaultSystemCodePage : Word
10857: Function ZCompatibleCodePages(const CP1, CP2 : Word) : Boolean
10858:
10859:
10860: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10861: begin
10862:   'RPC_C_AUTHN_LEVEL_DEFAULT', 'LongInt' ( 0);
10863:   ('RPC_C_AUTHN_LEVEL_NONE', 'LongInt' ( 1);
10864:   ('RPC_C_AUTHN_LEVEL_CONNECT', 'LongInt' ( 2);
10865:   ('RPC_C_AUTHN_LEVEL_CALL', 'LongInt' ( 3);
10866:   ('RPC_C_AUTHN_LEVEL_PKT', 'LongInt' ( 4);
10867:   ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY', 'LongInt' ( 5);
10868:   ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY', 'LongInt' ( 6);
10869:   {'alDefault', '1 RPC_C_AUTHN_LEVEL_DEFAULT');
10870:   ('alNone', '2 RPC_C_AUTHN_LEVEL_NONE);
10871:   ('alConnect', '3 RPC_C_AUTHN_LEVEL_CONNECT);
10872:   ('alCall', '4 RPC_C_AUTHN_LEVEL_CALL);
10873:   ('alPacket', '5 RPC_C_AUTHN_LEVEL_PKT);
10874:   ('alPacketIntegrity', '6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10875:   ('alPacketPrivacy', '7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10876:   ('RPC_C_IMP_LEVEL_DEFAULT', 'LongInt' ( 0);
10877:   ('RPC_C_IMP_LEVEL_ANONYMOUS', 'LongInt' ( 1);
10878:   ('RPC_C_IMP_LEVEL_IDENTIFY', 'LongInt' ( 2);
10879:   ('RPC_C_IMP_LEVEL_IMPERSONATE', 'LongInt' ( 3);
10880:   ('RPC_C_IMP_LEVEL_DELEGATE', 'LongInt' ( 4);
10881:   {'ilDefault', '0 RPC_C_IMP_LEVEL_DEFAULT);
10882:   ('ilAnonymous', '1 RPC_C_IMP_LEVEL_ANONYMOUS);
10883:   ('ilIdentiry', '2 RPC_C_IMP_LEVEL_IDENTIFY);
10884:   ('ilImpersonate', '3 RPC_C_IMP_LEVEL_IMPERSONATE);
10885:   ('ilDelegate', '4 RPC_C_IMP_LEVEL_DELEGATE);}
10886:   ('EOAC_NONE', 'LongWord').SetUInt( $0);
10887:   ('EOAC_DEFAULT', 'LongWord').SetUInt( $800);
10888:   ('EOAC_MUTUAL_AUTH', 'LongWord').SetUInt( $1);
10889:   ('EOAC_STATIC_CLOACKING', 'LongWord').SetUInt( $20);
10890:   ('EOAC_DYNAMIC_CLOACKING', 'LongWord').SetUInt( $40);
10891:   ('EOAC_ANY_AUTHORITY', 'LongWord').SetUInt( $80);
10892:   ('RPC_C_AUTHN_WINNT', 'LongInt' ( 10);
10893:   ('RPC_C_AUTHN_NONE', 'LongInt' ( 0);
10894:   ('RPC_C_AUTHN_NAME', 'LongInt' ( 1);
10895:   ('RPC_C_AUTHN_DCE', 'LongInt' ( 2);
10896:   FindClass('TOBJECT'), 'E-BoldCom
10897: Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10898: Function BoldMemoryToVariant(const Buffer, BufSize : Integer) : OleVariant
10899: Function BoldStreamToVariant( Stream : TStream) : OleVariant
10900: Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10901: Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10902: Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10903: Function BoldVariantArrayOfStringsToStrings( V : OleVariant; Strings : TStrings) : Integer
10904: Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10905: Function BoldCreateNamedValues(const Names:array of string; const Values:array of OleVariant):OleVariant;
10906: Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10907: Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10908: Function BoldCreateGUID : TGUID
10909: Function BoldCreateComObject(const ClsId, IId : TGUID; out Obj : variant; out Res : HRESULT) : Boolean
10910: Function BoldCreateRemoteComObject(const HostName:string; const ClsId, IId:TGUID; out Obj:variant; out
    Res:HRESULT):Bool;
10911: Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10912: Procedure BoldSetSecurityForInterface(AuthenticationLevel, ImpersonationLevel:longint;Unk:IUnknown);
10913: end;
10914:
10915: (*-----*)
10916: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10917: begin
10918:   Function ParseISODate( s : string) : TDateTime
10919:   Function ParseISODateTime( s : string) : TDateTime
10920:   Function ParseISOTime( str : string) : TDateTime
10921: end;
10922:
10923: (*-----*)
10924: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10925: begin
10926:   Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10927:   Function BoldCreateGUIDWithBracketsAsString : string
10928: end;
10929:
10930: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10931: begin
10932:   FindClass('TOBJECT'), 'T-BoldFileHandler
10933:   FindClass('TOBJECT'), 'T-BoldDiskFileHandler
10934:   //T-BoldFileHandlerClass', 'class of T-BoldFileHandler
10935:   T-BoldInitializeFileContents', 'Procedure ( StringList : TStringList)
10936:   SIRegister_TBoldFileHandler(CL);
10937:   SIRegister_TBoldDiskFileHandler(CL);
10938: Procedure BoldCloseAllFilehandlers

```

```

10939: Procedure BoldRemoveUnchangedFilesFromEditor
10940: Function BoldFileHandlerList : TBoldObjectArray
10941: Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
10942: end;
10943:
10944: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
10945: begin
10946:   PCharArr', 'array of PChar
10947:   Function BoldInternetOpen(Agent:String;
AccessType:integer;Proxy:String;ProxyByPass:String;Flags:integer):ptr);
10948:   Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
10949:   Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
NumberOfBytesRead:Card):LongBool;
10950:   Function BoldInternetCloseHandle( HINet : Pointer ) : LongBool
10951:   Function BoldHttpRequestInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
Cardinal; Reserved : Cardinal ) : LongBool
10952:   Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
Cardinal; Context : Cardinal ) : LongBool
10953:   Function BoldHttpRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
: PCharArr; Flags, Context : Cardinal ) : Pointer
10954:   Function BoldHttpRequestSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;Optionallength:Cardinal): LongBool
10955:   Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
10956:   Function BoldInternetAttemptConnect( dwReserved : DWORD ) : DWORD
10957:   Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
10958:   Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
10959: end;
10960:
10961: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
10962: begin
10963:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
10964:   SIRegister_TfrmBoldQueryUser(CL);
10965:   Function QueryUser( const Title, Query : string ) : TBoldQueryResult
10966: end;
10967:
10968: (*-----*)
10969: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
10970: begin
10971:   (('befIsInDisplayList',' BoldElementFlag0);
10972:   (('befStronglyDependedOfPrioritized',' BoldElementFlag1);
10973:   (('befFollowerSelected',' BoldElementFlag2);
10974:   FindClass('TOBJECT'),'TBoldQueue
10975:   FindClass('TOBJECT'),'TBoldQueueable
10976:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
10977:   SIRegister_TBoldQueueable(CL);
10978:   SIRegister_TBoldQueue(CL);
10979:   Function BoldQueueFinalized : Boolean
10980:   Function BoldInstalledQueue : TBoldQueue
10981: end;
10982:
10983: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
10984: begin
10985:   const mmPerInch,'Extended').setExtended( 25.4);
10986:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,
10987:   +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
10988:   +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
10989:   +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
10990:   +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
10991:   TBarLineType', '( white, black, black_half )
10992:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
10993:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
10994:   +'pBottomLeft, stpBottomRight, stpBottomCenter )
10995:   TChecksumMethod', '( csmNone, csmModulo10 )
10996:   SIRegister_TAsBarcode(CL);
10997:   Function CheckSumModulo10( const data : string ) : string
10998:   Function ConvertMmToPixelsX( const Value : Double ) : Integer
10999:   Function ConvertMmToPixelsY( const Value : Double ) : Integer
11000:   Function ConvertInchToPixelsX( const Value : Double ) : Integer
11001:   Function ConvertInchToPixelsY( const Value : Double ) : Integer
11002: end;
11003:
11004: procedure SIRegister_Geometry(CL: TPSPascalCompiler); //OpenGL
11005: begin
11006:   THomogeneousByteVector', 'array[0..3] of Byte
11007:   THomogeneousWordVector', 'array[0..3] of Word
11008:   THomogeneousIntVector', 'array[0..3] of Integer
11009:   THomogeneousFltVector', 'array[0..3] of single
11010:   THomogeneousDblVector', 'array[0..3] of double
11011:   THomogeneousExtVector', 'array[0..3] of extended
11012:   TAffineByteVector', 'array[0..2] of Byte
11013:   TAffineWordVector', 'array[0..2] of Word
11014:   TAffineIntVector', 'array[0..2] of Integer
11015:   TAffineFltVector', 'array[0..2] of single
11016:   TAffineDblVector', 'array[0..2] of double
11017:   TAffineExtVector', 'array[0..2] of extended
11018:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11019:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11020:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector

```

```

11021: THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11022: THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11023: THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11024: TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11025: TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11026: TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11027: TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11028: TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11029: TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11030: TMatrix4b', 'THomogeneousByteMatrix
11031: TMatrix4w', 'THomogeneousWordMatrix
11032: TMatrix4i', 'THomogeneousIntMatrix
11033: TMatrix4f', 'THomogeneousFltMatrix
11034: TMatrix4d', 'THomogeneousDblMatrix
11035: TMatrix4e', 'THomogeneousExtMatrix
11036: TMatrix3b', 'TAffineByteMatrix
11037: TMatrix3w', 'TAffineWordMatrix
11038: TMatrix3i', 'TAffineIntMatrix
11039: TMatrix3f', 'TAffineFltMatrix
11040: TMatrix3d', 'TAffineDblMatrix
11041: TMatrix3e', 'TAffineExtMatrix
11042: //PMatrix', '^TMatrix // will not work
11043: TMatrixGL', 'THomogeneousFltMatrix
11044: THomogeneousMatrix', 'THomogeneousFltMatrix
11045: TAffineMatrix', 'TAffineFltMatrix
11046: TQuaternion', 'record Vector : TVector4f; end
11047: TRectangle', 'record Left : integer; Top : integer; Width : inte'
11048: + 'ger; Height : Integer; end
11049: TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11050: + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11051: + ', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11052: 'EPSILON', 'Extended').setExtended( 1E-100);
11053: 'EPSILON2', 'Extended').setExtended( 1E-50);
11054: Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11055: Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11056: Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11057: Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11058: Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11059: Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11060: Function VectorAngle( V1, V2 : TAffineVector) : Single
11061: Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11062: Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11063: Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11064: Function VectorLength( V : array of Single) : Single
11065: Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11066: Procedure VectorNegate( V : array of Single)
11067: Function VectorNorm( V : array of Single) : Single
11068: Function VectorNormalize( V : array of Single) : Single
11069: Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11070: Function VectorReflect( V, N : TAffineVector) : TAffineVector
11071: Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11072: Procedure VectorScale( V : array of Single; Factor : Single)
11073: Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11074: Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11075: Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11076: Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11077: Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11078: Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11079: Procedure MatrixAdjoint( var M : TMatrixGL)
11080: Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11081: Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11082: Function MatrixDeterminant( M : TMatrixGL) : Single
11083: Procedure MatrixInvert( var M : TMatrixGL)
11084: Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11085: Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11086: Procedure MatrixTranspose( var M : TMatrixGL)
11087: Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11088: Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11089: Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11090: Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11091: Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11092: Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11093: Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11094: Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11095: //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11096: Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11097: Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11098: Function VectorTransforml( V : TVector3f; M : TMatrixGL) : TVector3f;
11099: Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11100: Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11101: Function MakeAffineVector( V : array of Single) : TAffineVector
11102: Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11103: Function MakeVector( V : array of Single) : TVectorGL
11104: Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11105: Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11106: Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11107: Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11108: Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11109: Function ArcCosGL( X : Extended) : Extended

```



```

11110: Function ArcSinGL( X : Extended) : Extended
11111: Function ArcTan2GL( Y, X : Extended) : Extended
11112: Function CoTanGL( X : Extended) : Extended
11113: Function DegToRadGL( Degrees : Extended) : Extended
11114: Function RadToDegGL( Radians : Extended) : Extended
11115: Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11116: Function TanGL( X : Extended) : Extended
11117: Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11118: Function Turnl( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11119: Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11120: Function Pitchl( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11121: Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11122: Function Rolll( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11123: end;
11124:
11125:
11126: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11127: begin
11128:   Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11129:   Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11130:   Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11131:   Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11132:   Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11133:   Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11134:   Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11135:   Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11136:   Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11137:   Function RegReadWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11138:   Function RegReadWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11139:   Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11140:   Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11141:   Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11142:   Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11143:   Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11144:   Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11145:   Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11146:   Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11147:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11148:     + 'RunOnce, ekServiceRun, ekServiceRunOnce )
11149:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11150:   Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11151:   Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11152:   Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
Items:TStrings):Bool;
11153:   Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
SaveTo:TStrings):Bool;
11154:   Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11155: end;
11156:
11157: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11158: begin
11159:   CLSID_StdComponentCategoriesMgr, 'TGUID '{0002E005-0000-0000-C000-000000000046}
11160:   CATID_SafeForInitializing, 'TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11161:   CATID_SafeForScripting, 'TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11162:   icMAX_CATEGORY_DESC_LEN, 'LongInt'( 128);
11163:   FindClass('TOBJECT'),'EInvalidParam
11164:   Function IsDCOMInstalled : Boolean
11165:   Function IsDCOMEnabled : Boolean
11166:   Function GetDCOMVersion : string
11167:   Function GetMDACVersion : string
11168:   Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11169:   Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11170:   Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11171:   Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11172:   Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11173:   Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11174:   Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11175:   Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11176:   Function ResetIStreamToStart( Stream : IStream) : Boolean
11177:   Function SizeOfIStreamContents( Stream : IStream) : Largeint
11178:   Function StreamToVariantArray( Stream : TStream) : OleVariant;
11179:   Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11180:   Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11181:   Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11182: end;
11183:
11184:
11185: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11186: begin
11187:   Const ('CelsiusFreezingPoint', 'Extended').setExtended( 0.0);
11188:   FahrenheitFreezingPoint, 'Extended').setExtended( 32.0);
11189:   KelvinFreezingPoint, 'Extended').setExtended( 273.15);
11190:   CelsiusAbsoluteZero, 'Extended').setExtended( - 273.15);
11191:   FahrenheitAbsoluteZero, 'Extended').setExtended( - 459.67);
11192:   KelvinAbsoluteZero, 'Extended').setExtended( 0.0);
11193:   DegPerCycle, 'Extended').setExtended( 360.0);

```

PC08 E:\maxbox3\mXGit39988\maxbox3\docs\maxbox_extract_funclist399.txt
<http://www.softwareschule.ch/maxbox.htm>

```

11283: Function HpMetricToWatt( const HpM : double) : double
11284: Function MsToKnot( const ms : double) : double
11285: Function WattToHpElectric( const W : double) : double
11286: Function WattToHpMetric( const W : double) : double
11287: function getBigPI: string; //PI of 1000 numbers
11288:
11289: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11290: begin
11291:   Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11292:   Procedure CDCopyFile( const FileName, DestName : string)
11293:   Procedure CDMoveFile( const FileName, DestName : string)
11294:   Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11295:   Procedure CDDeleteFiles( Sender : TObject; s : string)
11296:   Function CDGetTempDir : string
11297:   Function CDGetFileSize( FileName : string) : longint
11298:   Function GetFileTime( FileName : string) : longint
11299:   Function GetShortName( FileName : string) : string
11300:   Function GetFullName( FileName : string) : string
11301:   Function WinReboot : boolean
11302:   Function WinDir : String
11303:   Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11304:   Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11305:   Function devExecutor : TdevExecutor
11306: end;
11307:
11308: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11309: begin
11310:   Procedure CheckAssociations // AssociationsCount', 'LongInt'( 7);
11311:   Procedure Associate( Index : integer)
11312:   Procedure UnAssociate( Index : integer)
11313:   Function IsAssociated( Index : integer) : boolean
11314:   Function CheckFileType( const extension, filetype, description, verb, serverapp : string) : boolean
11315:   Procedure RegisterFileType( const extension, filetype, description, verb, serverapp, IconNum: string)
11316:   Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11317:   procedure RefreshIcons;
11318:   function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11319:   function MergColor(Colors: Array of TColor): TColor;
11320:   function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11321:   procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11322:   function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11323:   function GetInverseColor(AColor: TColor): TColor;
11324:   procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11325:   procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X,Y: integer; ShadowColor: TColor);
11326:   procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11327:   Procedure GetSystemMenuFont(Font: TFont);
11328: end;
11329:
11330: //*****unit uPSI_JvHLParse;*****
11331: function IsStringConstant(const St: string): Boolean;
11332: function IsIntConstant(const St: string): Boolean;
11333: function IsRealConstant(const St: string): Boolean;
11334: function IsIdentifier(const ID: string): Boolean;
11335: function GetStringValue(const St: string): string;
11336: procedure ParseString(const S: string; Ss: TStrings);
11337: function IsStringConstantW(const St: WideString): Boolean;
11338: function IsIntConstantW(const St: WideString): Boolean;
11339: function IsRealConstantW(const St: WideString): Boolean;
11340: function IsIdentifierW(const ID: WideString): Boolean;
11341: function GetStringValueW(const St: WideString): WideString;
11342: procedure ParseStringW(const S: WideString; Ss: TStrings);
11343:
11344:
11345: //*****unit uPSI_JclMapi;*****
11346:
11347: Function JclSimpleSendMail( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
  TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11348: Function JclSimpleSendFax( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
  TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11349: Function JclSimpleBringUpSendMailDialog(const ASubject, ABody: string; const
  AAttach: TFileName; AParentWND: HWND): Bool
11350: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11351: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11352:
11353: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11354: begin
11355:   // 'Pdes_key_schedule', '^des_key_schedule // will not work
11356:   Function BuildType1Message( ADomain, AHost : String) : String
11357:   Function BuildType3Message(ADomain, AHost, AUsername: WideString; APassword, ANonce: String): String
11358:   Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11359:   Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11360:   GBase64CodeTable', 'string' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11361:   GXXECodeTable', 'string' ++0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11362:   GUUECodeTable', 'string' '!\"#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_
11363: end;
11364:
11365: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11366: begin
11367:   ('IpAny', 'LongWord').SetUInt( $00000000);
11368:   IpLoopBack', 'LongWord').SetUInt( $7F000001);

```

```

11369: IpBroadcast', 'LongWord').SetUInt( $FFFFFFF);
11370: IpNone', 'LongWord').SetUInt( $FFFFFFF);
11371: PortAny', 'LongWord( $0000);
11372: SocketMaxConnections', 'LongInt'( 5);
11373: TIpAddr', 'LongWord
11374: TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11375: Function HostToNetLong( HostLong : LongWord) : LongWord
11376: Function HostToNetShort( HostShort : Word) : Word
11377: Function NetToHostLong( NetLong : LongWord) : LongWord
11378: Function NetToHostShort( NetShort : Word) : Word
11379: Function StrToIp( Ip : string) : TIpAddr
11380: Function IpToStr( Ip : TIpAddr) : string
11381: end;
11382:
11383: (*-----*)
11384: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11385: begin
11386:   TALSMTPClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11387:   + 'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11388:   + 'amShal, AlsmtpClientAuthAutoSelect )
11389:   TALSMTPClientAuthTypeSet', 'set of TALSMTPClientAuthType
11390:   SIRegister_TALSMTPClient(CL);
11391: end;
11392:
11393: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11394: begin
11395:   TBitNo', 'Integer
11396:   TStByteNo', 'Integer
11397:   TStationNo', 'Integer
11398:   TInOutNo', 'Integer
11399:   TIO', '( EE, AA, NE, NA )
11400:   TBitSet', 'set of TBitNo
11401:   TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11402:   TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11403:   TBitAddr', 'LongInt
11404:   TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11405:   TByteAddr', 'SmallInt
11406:   TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11407:   Function BitAddr(aIo: TIO; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11408:   Function BusBitAddr(aIo:TIO;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11409:   Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte;var
aBitNo:TBitNo);
11410:   Function BitAddrToStr( Value : TBitAddr) : string
11411:   Function StrToBitAddr( const Value : string) : TBitAddr
11412:   Function ByteAddr( aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11413:   Function BusByteAddr(aIo:TIO;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11414:   Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte)
11415:   Function ByteAddrToStr( Value : TByteAddr) : string
11416:   Function StrToByteAddr( const Value : string) : TByteAddr
11417:   Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11418:   Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11419:   Function InOutStateToStr( State : TInOutState) : string
11420:   Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11421:   Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11422: end;
11423:
11424: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11425: begin
11426:   TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11427:   + 'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11428:   DpmiPmVector', 'Int64
11429:   'DInterval', 'LongInt'( 1000);
11430:   // 'DEnabled', 'Boolean') BoolToStr( True);
11431:   'DIntFreq', 'string' if64
11432:   // 'DMessages', 'Boolean if64);
11433:   SIRegister_TwdxCustomTimer(CL);
11434:   SIRegister_TwdxTimer(CL);
11435:   SIRegister_TwdxRtcTimer(CL);
11436:   SIRegister_TCustomIntTimer(CL);
11437:   SIRegister_TIntTimer(CL);
11438:   SIRegister_TRtcIntTimer(CL);
11439:   Function RealNow : TDateTime
11440:   Function MsToTime( MilliSecond : LongInt) : TDateTime
11441:   Function DateTimeToMs( Time : TDateTime) : LongInt
11442: end;
11443:
11444: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11445: begin
11446:   TIdSyslogPRI', 'Integer
11447:   TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11448:   + 'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTTP, sfUUCP, sfCloc'
11449:   + 'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11450:   + 'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11451:   + 'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11452:   TIdSyslogSeverity', '(slEmergency, slAlert, slCritical, slError, slWarning, slNotice, slInformational, slDebug)
11453:   SIRegister_TIdSysLogMsgPart(CL);
11454:   SIRegister_TIdSysLogMessage(CL);
11455:   Function FacilityToString( AFac : TIdSyslogFacility) : string
11456:   Function SeverityToString( ASec : TIdSyslogSeverity) : string

```



```

11457: Function NoToSeverity( ASev : Word ) : TIdSyslogSeverity
11458: Function logSeverityToNo( ASev : TIdSyslogSeverity ) : Word
11459: Function NoToFacility( AFac : Word ) : TIdSyslogFacility
11460: Function logFacilityToNo( AFac : TIdSyslogFacility ) : Word
11461: end;
11462:
11463: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11464: begin
11465:   'UWhitespace','String' '(?:\s*)
11466:   Function StripSpaces( const AText : string ) : string
11467:   Function CharCount( const AText : string; Ch : Char ) : Integer
11468:   Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11469:   Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11470: end;
11471:
11472:
11473: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11474: begin
11475:   ExtPascalVersion,'String' '0.9.8
11476:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11477:     + 'Opera, brKonqueror, brMobileSafari )
11478:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11479:   AddTypeS('TExtProcedure', 'Procedure
11480:   Function DetermineBrowser( const UserAgentStr : string ) : TBrowser
11481:   Function ExtExtract(const Delims:array of string; var S:string; var Matches:TStringList; Remove:bool):bool;
11482:   Function ExtExplode( Delim : char; const S : string; Separator : char ) : TStringList
11483:   Function FirstDelimiter( const Delimiters, S : string; Offset : integer ) : integer
11484:   Function RPosEx( const Substr, Str : string; Offset : integer ) : integer
11485:   Function CountStr( const Substr, Str : string; UntilStr : string ) : integer
11486:   Function StrToJS( const S : string; UseBR : boolean ) : string
11487:   Function CaseOf( const S : string; const Cases : array of string ) : integer
11488:   Function RCaseOf( const S : string; const Cases : array of string ) : integer
11489:   Function EnumToJSSString( TypeInfo : PTypeInfo; Value : integer ) : string
11490:   Function SetPaddings(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11491:   Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11492:   Function ExtBefore( const BeforeS, AfterS, S : string ) : boolean
11493:   Function IsUpperCase( S : string ) : boolean
11494:   Function BeautifyJS(const AScript:string; const StartingLevel:integer; SplitHTMLNewLine: boolean):string;
11495:   Function BeautifyCSS( const AStyle : string ) : string
11496:   Function LengthRegExp( Rex : string; CountAll : Boolean ) : integer
11497:   Function JSDateToDateTime( JSDate : string ) : TDateTime
11498: end;
11499:
11500: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11501: begin
11502:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11503:   TSHDeleteOptions', 'set of TSHDeleteOption
11504:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11505:   TSHRenameOptions', 'set of TSHRenameOption
11506:   Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions ) : Boolean
11507:   Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions ) : Boolean
11508:   Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions ) : Boolean
11509:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11510:   TEnumFolderFlags', 'set of TEnumFolderFlag
11511:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWORD'
11512:     + 'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIDList; EnumIDList : '
11513:     + 'IEnumIDList; Folder : IShellFolder; end
11514:   Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11515:   Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11516:   Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11517:   Function SHEnumFolderNext( var F : TEnumFolderRec ) : Boolean
11518:   Function GetSpecialFolderLocation( const Folder : Integer ) : string
11519:   Function DisplayPropDialog( const Handle : HWND; const FileName : string ) : Boolean;
11520:   Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIDList ) : Boolean;
11521:   Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint ) : Boolean
11522:   Function OpenFolder( const Path : string; Parent : HWND ) : Boolean
11523:   Function OpenSpecialFolder( FolderID : Integer; Parent : HWND ) : Boolean
11524:   Function SHReallocMem( var P : Pointer; Count : Integer ) : Boolean
11525:   Function SHAllocMem( out P : Pointer; Count : Integer ) : Boolean
11526:   Function SHGetMem( var P : Pointer; Count : Integer ) : Boolean
11527:   Function SHFreeMem( var P : Pointer ) : Boolean
11528:   Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder ) : PItemIDList
11529:   Function PathToPidl( const Path : string; Folder : IShellFolder ) : PItemIDList
11530:   Function PathToPidlBind( const FileName : string; out Folder : IShellFolder ) : PItemIDList
11531:   Function PidlBindToParent(const IdList:PItemIDList;out Folder:IShellFolder;out Last:PItemIDList):Bool;
11532:   Function PidlCompare( const Pidl1, Pidl2 : PItemIDList ) : Boolean
11533:   Function PidlCopy( const Source : PItemIDList; out Dest : PItemIDList ) : Boolean
11534:   Function PidlFree( var IdList : PItemIDList ) : Boolean
11535:   Function PidlGetDepth( const Pidl : PItemIDList ) : Integer
11536:   Function PidlGetLength( const Pidl : PItemIDList ) : Integer
11537:   Function PidlGetNext( const Pidl : PItemIDList ) : PItemIDList
11538:   Function PidlToPath( IdList : PItemIDList ) : string
11539:   Function StrRetFreeMem( StrRet : TStrRet ) : Boolean
11540:   Function StrRetToString( IdList : PItemIDList; StrRet : TStrRet; Free : Boolean ) : string
11541:   PShellLink', '^TShellLink // will not work
11542:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11543:     + 'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11544:     + ' : string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11545:   Procedure ShellLinkFree( var Link : TShellLink)

```

```

11546: Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11547: Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11548: Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11549: Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11550: Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11551: Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11552: Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11553: Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11554: Function OverlayIconShared( var Large, Small : HICON) : Boolean
11555: Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11556: Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11557: Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11558: Function ShellExecAndWait(const FileName:string;const Params:string;const Verb:string;CmdShow:Int):Bool;
11559: Function ShellOpenAs( const FileName : string) : Boolean
11560: Function ShellRasDial( const EntryName : string) : Boolean
11561: Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11562: Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11563: TJclFileExeType, '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11564: Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11565: Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11566: Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11567: Function OemKeyScan( wOemChar : Word) : DWORD
11568: Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11569: end;
11570:
11571: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11572: begin
11573:   xmlVersion','String '1.0 FindClass('OBJECT'),'Exml
11574:   //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11575:   Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11576:   Function xmlValidChar2( const Ch : WideChar) : Boolean;
11577:   Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11578:   Function xmlIsLetter( const Ch : WideChar) : Boolean
11579:   Function xmlIsDigit( const Ch : WideChar) : Boolean
11580:   Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11581:   Function xmlIsNameChar( const Ch : WideChar) : Boolean
11582:   Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11583:   Function xmlValidName( const Text : UnicodeString) : Boolean
11584:   //xmlSpace','Char #$20 or #$9 or #$D or #$A);
11585:   //Function xmlSkipSpace( var P : PWideChar) : Boolean
11586:   //Function xmlSkipEq( var P : PWideChar) : Boolean
11587:   //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11588:   //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
: TUnicodeCodecClass
11589: Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11590: Function xmlTag( const Tag : UnicodeString) : UnicodeString
11591: Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11592: Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11593: Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11594: Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11595: Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11596: Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11597: Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11598: Function xmlComment( const Comment : UnicodeString) : UnicodeString
11599: Procedure SelfTestcXMLFunctions
11600: end;
11601:
11602: (*-----*)
11603: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11604: begin
11605:   Function AWaitCursor : IUnknown
11606:   Function ChangeCursor( NewCursor : TCursor) : IUnknown
11607:   Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11608:   Function YesNo( const ACaption, AMsg : string) : boolean
11609:   Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11610:   Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11611:   Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11612:   Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11613:   Procedure GetSystemPaths( Strings : TStrings)
11614:   Procedure MakeEditNumeric( EditHandle : integer)
11615: end;
11616:
11617: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11618: begin
11619:   AddTypeS('TVideoCodec', '(vcUnknown,vcRGB,vcUYU2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11620:   'BI_UYU2','LongWord( $32595559);
11621:   'BI_UYVY','LongWord').SetUInt( $59565955);
11622:   'BI_BTUV','LongWord').SetUInt( $50313459);
11623:   'BI_YVU9','LongWord').SetUInt( $39555659);
11624:   'BI_YUV12','LongWord( $30323449);
11625:   'BI_Y8','LongWord').SetUInt( $20203859);
11626:   'BI_Y211','LongWord').SetUInt( $31313259);
11627:   Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11628:   Function ConvertCodecToRGB(Coec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11629: end;
11630:
11631: (*-----*)
11632: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11633: begin

```

```

11634: 'WM_USER', 'LongWord').SetUInt( $0400);
11635: 'WM_CAP_START', 'LongWord').SetUInt($0400);
11636: 'WM_CAP_END', 'longword').SetUInt($0400+85);
11637: //WM_CAP_START+ 85
11638: // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11639: Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11640: Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11641: Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11642: Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11643: Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11644: Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11645: Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11646: Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11647: Function capGetUserData( hwnd : THandle) : LongInt
11648: Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11649: Function capDriverDisconnect( hwnd : THandle) : LongInt
11650: Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11651: Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11652: Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11653: Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11654: Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11655: Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11656: Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11657: Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11658: Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11659: Function capEditCopy( hwnd : THandle) : LongInt
11660: Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11661: Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11662: Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11663: Function capDlgVideoFormat( hwnd : THandle) : LongInt
11664: Function capDlgVideoSource( hwnd : THandle) : LongInt
11665: Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11666: Function capDlgVideoCompression( hwnd : THandle) : LongInt
11667: Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11668: Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11669: Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11670: Function capPreview( hwnd : THandle; f : Word) : LongInt
11671: Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11672: Function capOverlay( hwnd : THandle; f : Word) : LongInt
11673: Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11674: Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11675: Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11676: Function capGrabFrame( hwnd : THandle) : LongInt
11677: Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11678: Function capCaptureSequence( hwnd : THandle) : LongInt
11679: Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11680: Function capCaptureStop( hwnd : THandle) : LongInt
11681: Function capCaptureAbort( hwnd : THandle) : LongInt
11682: Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11683: Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11684: Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11685: Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11686: Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11687: Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11688: Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11689: Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11690: Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11691: Function capPalettePaste( hwnd : THandle) : LongInt
11692: Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11693: Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11694: //PCapDriverCaps', '^TCapDriverCaps // will not work
11695: TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11696: +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11697: +'; y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11698: +'; eoin : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11699: //PCapStatus', '^TCapStatus // will not work
11700: TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11701: +'; fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOINT'
11702: +'; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11703: +'; OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11704: +'; rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11705: +'; ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11706: +'; wNumAudioAllocated : WORD; end
11707: //PCaptureParms', '^TCaptureParms // will not work
11708: TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11709: +'; UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11710: +'; ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11711: +'; deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11712: +'; Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11713: +'; ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11714: +'; wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11715: +'; epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11716: +'; he : BOOL; AVStreamMaster : WORD; end
11717: // PCapInfoChunk', '^TCapInfoChunk // will not work
11718: //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11719: 'CONTROLCALLBACK_PREROLL', 'LongInt'( 1);
11720: 'CONTROLCALLBACK_CAPTURING', 'LongInt'( 2);
11721: Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
: Integer; hwndParent : THandle; nHD : Integer) : THandle

```

```

11722: Function
      capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11723: 'IDS_CAP_BEGIN','LongInt'( 300);
11724: 'IDS_CAP_END','LongInt'( 301);
11725: 'IDS_CAP_INFO','LongInt'( 401);
11726: 'IDS_CAP_OUTOFMEM','LongInt'( 402);
11727: 'IDS_CAP_FILEEXISTS','LongInt'( 403);
11728: 'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11729: 'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11730: 'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11731: 'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11732: 'IDS_CAP_DEFPALEXT','LongInt'( 408);
11733: 'IDS_CAP_CANTOPEN','LongInt'( 409);
11734: 'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11735: 'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11736: 'IDS_CAP_VIDEDITERR','LongInt'( 412);
11737: 'IDS_CAP_READONLYFILE','LongInt'( 413);
11738: 'IDS_CAP_WRITEERROR','LongInt'( 414);
11739: 'IDS_CAP_NODISKSPACE','LongInt'( 415);
11740: 'IDS_CAP_SETFILESIZE','LongInt'( 416);
11741: 'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11742: 'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11743: 'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11744: 'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11745: 'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11746: 'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11747: 'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11748: 'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11749: 'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11750: 'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11751: 'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11752: 'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11753: 'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11754: 'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11755: 'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11756: 'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11757: 'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11758: 'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11759: 'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11760: 'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11761: 'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11762: 'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11763: 'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11764: 'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11765: 'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11766: 'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11767: 'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11768: 'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11769: 'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11770: 'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11771: 'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11772: 'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11773: 'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11774: 'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11775: 'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11776: 'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11777: 'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11778: 'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11779: 'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11780: 'AVICAP32','String 'AVICAP32.dll
11781: end;
11782:
11783: procedure SIRegister_ALFncMisc(CL: TPSPascalCompiler);
11784: begin
11785:   Function AlBoolToInt( Value : Boolean) : Integer
11786:   Function AlMediumPos( LTotal, LBorder, LObject : integer) : Integer
11787:   Function AlIsValidEmail( const Value : AnsiString) : boolean
11788:   Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TDateTime
11789:   Function AlInc( var x : integer; Count : integer) : Integer
11790:   function AlCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11791:   function AlGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11792:   procedure AlSaveStringToFile(Str: AnsiString; filename: AnsiString);
11793:   function AlIsInteger(const S: AnsiString): Boolean;
11794:   function AlIsDecimal(const S: AnsiString): boolean;
11795:   Function AlStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11796:   function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11797:   function AlQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11798:   function AlDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11799:   function AlUTF8removeBOM(const S: AnsiString): AnsiString;
11800:   Function AlRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11801:   Function AlRandomStr(const aLength: Longint): AnsiString;
11802:   Function AlRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11803:   Function AlRandomStrU(const aLength: Longint): String;
11804: end;
11805:
11806: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11807: begin
11808:   Procedure AlJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
aTrueStr: AnsiString; const aFalseStr : AnsiString)

```



```

11809: end;
11810:
11811: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11812: begin
11813:   _ALMEMORYSTATUSEX, 'record dwLength : DWORD; dwMemoryLoad : DWO'
11814:   + 'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11815:   + 'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11816:   + '; ullAvailExtendedVirtual : Int64; end
11817:   TALMemoryStatusEx, '_ALMEMORYSTATUSEX
11818:   Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX ) : BOOL
11819:   Function ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG ) : LONGLONG
11820:   'INVALID_SET_FILE_POINTER', 'LongInt'( DWORD ( - 1 ));
11821:   'QUOTA_LIMITS_HARDWS_MIN_DISABLE', 'LongWord').SetUInt( $2);
11822:   'QUOTA_LIMITS_HARDWS_MIN_ENABLE', 'LongWord').SetUInt( $1);
11823:   'QUOTA_LIMITS_HARDWS_MAX_DISABLE', 'LongWord').SetUInt( $8);
11824:   'QUOTA_LIMITS_HARDWS_MAX_ENABLE', 'LongWord').SetUInt( $4);
11825: end;
11826:
11827: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11828: begin
11829:   SIRegister_THandledObject(CL);
11830:   SIRegister_TEvent(CL);
11831:   SIRegister_TMutex(CL);
11832:   SIRegister_TSharedMem(CL);
11833:   'TRACE_BUF_SIZE', 'LongInt'( 200 * 1024);
11834:   'TRACE_BUFFER', 'String 'TRACE_BUFFER
11835:   'TRACE_MUTEX', 'String 'TRACE_MUTEX
11836:   //PTraceEntry, '^TTraceEntry // will not work
11837:   SIRegister_TIPCTracer(CL);
11838:   'MAX_CLIENTS', 'LongInt'( 6);
11839:   'IPCTIMEOUT', 'LongInt'( 2000);
11840:   'IPCBUFFER_NAME', 'String 'BUFFER_NAME
11841:   'BUFFER_MUTEX_NAME', 'String 'BUFFER_MUTEX
11842:   'MONITOR_EVENT_NAME', 'String 'MONITOR_EVENT
11843:   'CLIENT_EVENT_NAME', 'String 'CLIENT_EVENT
11844:   'CONNECT_EVENT_NAME', 'String 'CONNECT_EVENT
11845:   'CLIENT_DIR_NAME', 'String 'CLIENT_DIRECTORY
11846:   'CLIENT_DIR_MUTEX', 'String 'DIRECTORY_MUTEX
11847:   FindClass('TOBJECT'), 'EMonitorActive
11848:   FindClass('TOBJECT'), 'TIPCThread
11849:   TEventKind, '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11850:   + 'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11851:   + 'ach, evClientSwitch, evClientSignal, evClientExit )
11852:   TClientFlag, '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11853:   TClientFlags, 'set of TClientFlag
11854:   //PEventData, '^TEventData // will not work
11855:   TEventData, 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11856:   + 'lag; Flags : TClientFlags; end
11857:   TConnectEvent, 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11858:   TDirUpdateEvent, 'Procedure ( Sender : TIPCThread)
11859:   TIPCNofityEvent, 'Procedure ( Sender : TIPCThread; Data : TEventData)
11860:   //PIPCEventInfo, '^TIPCEventInfo // will not work
11861:   TIPCEventInfo, 'record FID: Integer; FKind: TEventKind; FData: TEventData; end
11862:   SIRegister_TIPCEvent(CL);
11863:   //PClientDirRecords, '^TClientDirRecords // will not work
11864:   SIRegister_TClientDirectory(CL);
11865:   TIPCState, '( stInActive, stDisconnected, stConnected )
11866:   SIRegister_TIPCThread(CL);
11867:   SIRegister_TIPCMonitor(CL);
11868:   SIRegister_TIPCClient(CL);
11869:   Function IsMonitorRunning( var Hndl : THandle ) : Boolean
11870: end;
11871:
11872: (*-----*)
11873: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11874: begin
11875:   SIRegister_TALGSMComm(CL);
11876:   Function ALGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString ) : AnsiString
11877:   Procedure ALGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
11878:   AMessage:AnsiString);
11879:   Function ALGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString ) : AnsiString
11879:   Function ALGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
11880:   UseGreekAlphabet:Bool):Widestring;
11880:   function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11881:   end;
11882:
11883: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11884: begin
11885:   TALHTTPPropertyChangeEvent, 'Procedure(sender:Tobject;const PropertyIndex:Integer;
11886:   TALHTTPProtocolVersion, '( HTTPPv_1_0, HTTPPv_1_1 )
11887:   TALHTTPMethod, '( HTTPmt_Get, HTTPmt_Post, HTTPmt_Head, HTTPmt_Trace, HTTPmt_Put, HTTPmt_Delete);
11888:   TInternetScheme, 'integer
11889:   TALIPv6Binary, 'array[1..16] of Char;
11890:   // TALIPv6Binary = array[1..16] of ansiChar;
11891:   // TInternetScheme = Integer;
11892:   SIRegister_TALHTTPRequestHeader(CL);
11893:   SIRegister_TALHTTPCookie(CL);
11894:   SIRegister_TALHTTPCookieCollection(CL);
11895:   SIRegister_TALHTTPResponseHeader(CL);

```

```

11896: Function ALHTTPDecode( const AStr : AnsiString ) : AnsiString
11897: Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11898: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
11899: // Procedure ALExtractHeaderFields( Separators, WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
11900: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators, WhiteSpace,
11901: // Quotes:TSysCharSet; Content:PAnsiChar; Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11901: Function ALRemoveSchemeFromUrl( aUrl : AnsiString ) : AnsiString
11902: Function ALExtractSchemeFromUrl( aUrl : AnsiString ) : TInternetScheme
11903: Function ALExtractHostNameFromUrl( aUrl : AnsiString ) : AnsiString
11904: Function ALExtractDomainNameFromUrl( aUrl : AnsiString ) : AnsiString
11905: Function ALExtractUrlPathFromUrl( aUrl : AnsiString ) : AnsiString
11906: Function ALInternetCrackUrl( aUrl : AnsiString; var SchemeName, HostName, UserName, Password, UrlPath,
11907: ExtraInfo : AnsiString; var PortNumber : integer ) : Boolean;
11907: Function ALInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password, UrlPath,
11908: Anchor : AnsiString; Query : TALStrings; var PortNumber : integer ) : Boolean;
11908: Function ALInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11909: Function ALRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString ) : AnsiString;
11910: Function ALRemoveAnchorFromUrl1( aUrl : AnsiString ) : AnsiString;
11911: Function ALCombineUrl( RelativeUrl, BaseUrl : AnsiString ) : AnsiString;
11912: Function ALCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11913: Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
11914: Function ALDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
11915: Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime ) : Boolean
11916: Function ALRfc822StrToGMTDateTime( const s : AnsiString ) : TDateTime
11917: Function ALTryIPv4StrToNumeric( aIPv4Str : AnsiString; var aIPv4Num : Cardinal ) : Boolean
11918: Function ALIPv4StrToNumeric( aIPv4 : AnsiString ) : Cardinal
11919: Function ALNumericToIPv4Str( aIPv4 : Cardinal ) : AnsiString
11920: Function ALZeroIPv6 : TALIPv6Binary
11921: Function ALTryIPv6StrToBinary( aIPv6Str : AnsiString; var aIPv6Bin : TALIPv6Binary ) : Boolean
11922: Function ALIPv6StrToBinary( aIPv6 : AnsiString ) : TALIPv6Binary
11923: Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary ) : AnsiString
11924: Function ALBinaryStrToIPv6Binary( aIPv6BinaryStr : AnsiString ) : TALIPv6Binary
11925: end;
11926:
11927: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11928: begin
11929: Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
11930: DecodeHTMLText:Bool;
11931: Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11932: Function ALXMLCDATAElementEncode( Src : AnsiString ) : AnsiString
11933: Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean ) : AnsiString
11934: Function ALUTF8XMLTextElementDecode( const Src : AnsiString ) : AnsiString
11935: Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIShtmlEntities:Bool;const
11936: useNumRef:bool):AnsiString);
11937: Function ALUTF8HTMLDecode( const Src : AnsiString ) : AnsiString
11938: Function ALJavaScriptEncode( const Src : AnsiString; const useNumericReference : boolean ) : AnsiString
11939: Function ALUTF8JavascriptDecode( const Src : AnsiString ) : AnsiString
11940: Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunc(var HtmlContent:AnsiString; const
11941: DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
11942: Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
11943: end;
11944:
11945: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11946: begin
11947: SIRegister_TALEmailHeader(CL);
11948: SIRegister_TALNewsArticleHeader(CL);
11949: Function ALParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AStr;const
11950: decodeRealName:Bool):AnsiString;
11951: Function ALExtractEmailAddress( FriendlyEmail : AnsiString ) : AnsiString
11952: Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString ) : AnsiString
11953: Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString ) : AnsiString
11954: Function ALGenerateInternetMessageID : AnsiString;
11955: Function ALGenerateInternetMessageID1( ahostname : AnsiString ) : AnsiString;
11956: Function ALDecodeQuotedPrintableString( src : AnsiString ) : AnsiString
11957: Function ALDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
11958: end;
11959:
11960: (*-----*)
11961: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
11962: begin
11963: Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
11964: Function ALIPAddrToName( IPAddr : AnsiString ) : AnsiString
11965: Function ALGetLocalIPs : TALStrings
11966: Function ALGetLocalHostName : AnsiString
11967: end;
11968:
11969: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
11970: begin
11971: Procedure ALCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
11972: TALWebRequest; ServerVariables:TALStrings);
11973: Procedure ALCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
11974: TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11975: Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
11976: Procedure ALCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
11977: ScriptFileName:AnsiString;Url:AnsiStr;
11978: Procedure ALCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
11979: ScriptName, ScriptFileName : AnsiString; Url : AnsiString);

```

```

11972: Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
: Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
11973: Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
WebRequest : TALIsapiRequest;
overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
11974: +'overloadedRequestContentStream:Tstream;var
ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
11975: Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
ResponseHeader : TALHTTPResponseHeader);
11976: end;
11977:
11978: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
11979: begin
11980:   TStartupInfoA', 'TStartupInfo
11981:   'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
11982:   SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
11983:   SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
11984:   SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
11985:   SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
11986:   SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
11987:   SE_TCB_NAME','String 'SeTcbPrivilege
11988:   SE_SECURITY_NAME','String 'SeSecurityPrivilege
11989:   SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
11990:   SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
11991:   SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
11992:   SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
11993:   SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
11994:   SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
11995:   SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
11996:   SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
11997:   SE_BACKUP_NAME','String 'SeBackupPrivilege
11998:   SE_RESTORE_NAME','String 'SeRestorePrivilege
11999:   SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12000:   SE_DEBUG_NAME','String 'SeDebugPrivilege
12001:   SE_AUDIT_NAME','String 'SeAuditPrivilege
12002:   SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12003:   SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12004:   SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12005:   SE_UNDOCK_NAME','String 'SeUndockPrivilege
12006:   SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12007:   SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12008:   SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12009:   Function AlGetEnvironmentString : AnsiString
12010:   Function ALWinExec32(const FileName,CurrentDir,
Environment:AnsiString;InStream:Tstream;OutStream:Tstream):Dword;
12011:   Function ALWinExec32l(const FileName:AnsiString; InputStream:Tstream;OutputStream:Tstream):Dword;
12012:   Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12013:   Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12014:   Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12015: end;
12016:
12017: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12018: begin
12019:   Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12020:   Function AlEmptyDirectoryl( Directory : ansiString; SubDirectory : Boolean; const
RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12021:   Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12022:   Function AlGetModuleName : ansiString
12023:   Function AlGetModuleFileNameWithoutExtension : ansiString
12024:   Function AlGetModulePath : ansiString
12025:   Function AlGetFileSize( const AFileName : ansiString) : int64
12026:   Function AlGetFileVersion( const AFileName : ansiString) : ansiString
12027:   Function AlGetFileCreationDateTime( const aFileName : AnsiString) : TdateTime
12028:   Function AlGetFileLastWriteDateTime( const aFileName : AnsiString) : TdateTime
12029:   Function AlGetFileLastAccessDateTime( const aFileName : AnsiString) : TdateTime
12030:   Procedure ALSetFileCreationDateTime( const aFileName : AnsiString; const aCreationDateTime : TdateTime)
12031:   Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12032:   Function ALFileExists( const Path : ansiString) : boolean
12033:   Function ALDirectoryExists( const Directory : AnsiString) : Boolean
12034:   Function ALCreateDir( const Dir : AnsiString) : Boolean
12035:   Function ALRemoveDir( const Dir : AnsiString) : Boolean
12036:   Function ALDeleteFile( const FileName : AnsiString) : Boolean
12037:   Function ALRenameFile( const OldName, NewName : ansiString) : Boolean
12038: end;
12039:
12040: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12041: begin
12042:   NativeInt', 'Integer
12043:   NativeUInt', 'Cardinal
12044:   Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12045:   Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12046:   Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12047:   Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12048:   Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12049:   Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12050:   Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)

```

```

12051: Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12052: Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12053: Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12054: Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;
+ 'var ByteBuffer : Cardinal; var ByteBufferSize : Cardinal) : NativeInt;
12056: Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
ByteBuffer : Cardinal; const ByteBufferSize : Cardinal) : NativeInt;
12057: Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
OutputBuf:TByteDynArray);
12058: Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12059: Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12060: Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray):NativeInt;
12061: Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSize : Cardinal) : NativeInt;
12062: Function ALMimeBase64DecodePartialEnd(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
ByteBufferSpace:Cardinal):NativeInt;
12063: Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12064: Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12065: Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12066: Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12067: Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12068: Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12069: 'cALMimeBase64_ENCODED_LINE_BREAK', 'LongInt' ( 76);
12070: 'cALMimeBase64_DECODED_LINE_BREAK', 'LongInt' ( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12071: 'cALMimeBase64_BUFFER_SIZE', 'LongInt' ( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12072: Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12073: Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12074: Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12075: Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12076: end;
12077:
12078: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12079: begin
12080: 'cALXMLNodeMaxListSize', 'LongInt' ( Maxint div 16);
12081: FindClass('TOBJECT'), 'TALXMLNode
12082: FindClass('TOBJECT'), 'TALXMLNodeList
12083: FindClass('TOBJECT'), 'TALXMLDocument
12084: TALXMLParseProcessingInstructionEvent', 'Procedure (Sender:TObject; const Target,Data:AnsiString)
12085: TALXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12086: TALXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12087: + 'nt Name : AnsiString; const Attributes : TALStrings)
12088: TALXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12089: TALXMLNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12090: + 'ntCDATA, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12091: + 'ntDocType, ntDocFragment, ntNotation )
12092: TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12093: TALXMLDocOptions', 'set of TALXMLDocOption
12094: TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12095: TALXMLParseOptions', 'set of TALXMLParseOption
12096: TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12097: PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12098: SIRegister_EALXMLDocError(CL);
12099: SIRegister_TALXMLNodeList(CL);
12100: SIRegister_TALXMLNode(CL);
12101: SIRegister_TALXmlElementNode(CL);
12102: SIRegister_TALXmlAttributeNode(CL);
12103: SIRegister_TALXmlTextNode(CL);
12104: SIRegister_TALXmlDocumentNode(CL);
12105: SIRegister_TALXmlCommentNode(CL);
12106: SIRegister_TALXmlProcessingInstrNode(CL);
12107: SIRegister_TALXmlCDATADataNode(CL);
12108: SIRegister_TALXmlEntityRefNode(CL);
12109: SIRegister_TALXmlEntityNode(CL);
12110: SIRegister_TALXmlDocTypeNode(CL);
12111: SIRegister_TALXmlDocFragmentNode(CL);
12112: SIRegister_TALXmlNotationNode(CL);
12113: SIRegister_TALXMLDocument(CL);
12114: cALXMLUTF8EncodingStr', 'String 'UTF-8
12115: cALXMLUTF8HeaderStr', 'String'<?xml version="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12116: CALNSDelim', 'String '
12117: CALXML', 'String 'xml
12118: CALVersion', 'String 'version
12119: CALEncoding', 'String 'encoding
12120: CALStandalone', 'String 'standalone
12121: CALDefaultNodeIndent', 'String '
12122: CALXmlDocument', 'String 'DOCUMENT
12123: Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12124: Procedure ALClearXMLDocument(const rootname:AnsiString;xmlDoc:TalXMLDocument;const
EncodingStr:AnsiString);
12125: Function ALFindXmlNodeByChildNodeValue(xmlrec:TalXmlNode;const ChildNodeName,
ChildNodeValue:AnsiString;const Recurse : Boolean) : TalXmlNode
12126: Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalXmlNode; const NodeName : ansiString; const
ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalXmlNode

```



```

12127: Function ALFindXmlNodeByAttribute(xmlrec: TalxmlNode; const AttributeName, AttributeValue: AnsiString; const
Recurse: Boolean): TalxmlNode
12128: Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12129: Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
AnsiString
12130: end;
12131:
12132: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12133: //based on TEEProc, TeCanvas, TEEEngine, TChart
12134: begin
12135:   'TeePiStep', 'Double').setExtended( Pi / 180.0);
12136:   'TeeDefaultPerspective', 'LongInt'( 100);
12137:   'TeeMinAngle', 'LongInt'( 270);
12138:   'teeclMoneyGreen', 'LongWord').SetUInt( TColor ( $C0DCC0 ));
12139:   'teeclSkyBlue', 'LongWord').SetUInt( TColor ( $FOCAA6 ));
12140:   'teeclCream', 'LongWord( TColor ( $F0FBFF ));
12141:   'teeclMedGray', 'LongWord').SetUInt( TColor ( $A4A0A0 ));
12142:   'teeclMoneyGreen', 'LongWord').SetUInt( TColor ( $C0DCC0 ));
12143:   'teeclSkyBlue', 'LongWord').SetUInt( TColor ( $FOCAA6 ));
12144:   'teeclCream', 'LongWord').SetUInt( TColor ( $F0FBFF ));
12145:   'teeclMedGray', 'LongWord').SetUInt( TColor ( $A4A0A0 ));
12146:   'TA_LEFT', 'LongInt'( 0);
12147:   'TA_RIGHT', 'LongInt'( 2);
12148:   'TA_CENTER', 'LongInt'( 6);
12149:   'TA_TOP', 'LongInt'( 0);
12150:   'TA_BOTTOM', 'LongInt'( 8);
12151:   'teePATCOPY', 'LongInt'( 0);
12152:   'NumCirclePoints', 'LongInt'( 64);
12153:   'teeDEFAULT_CHARSET', 'LongInt'( 1);
12154:   'teeANTIALIASED_QUALITY', 'LongInt'( 4);
12155:   'TA_LEFT', 'LongInt'( 0);
12156:   'bs_Solid', 'LongInt'( 0);
12157:   'teePf24Bit', 'LongInt'( 0);
12158:   'teePfDevice', 'LongInt'( 1);
12159:   'CM_MOUSELEAVE', 'LongInt'( 10000);
12160:   'CM_SYSCOLORCHANGE', 'LongInt'( 10001);
12161:   'DC_BRUSH', 'LongInt'( 18);
12162:   'DC_PEN', 'LongInt'( 19);
12163:   teeCOLORREF', 'LongWord
12164:   TLogBrush', record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12165:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12166:   SIRegister_TFilterRegion(CL);
12167:   SIRegister_IFormCreator(CL);
12168:   SIRegister_TTeeFilter(CL);
12169:   //TFilterClass', 'class of TTeeFilter
12170:   SIRegister_TFilterItems(CL);
12171:   SIRegister_TConvolveFilter(CL);
12172:   SIRegister_TBlurFilter(CL);
12173:   SIRegister_TTeePicture(CL);
12174:   TPenEndStyle', '( esRound, esSquare, esFlat )
12175:   SIRegister_TChartPen(CL);
12176:   SIRegister_TChartHiddenPen(CL);
12177:   SIRegister_TDottedGrayPen(CL);
12178:   SIRegister_TDarkGrayPen(CL);
12179:   SIRegister_TWhitePen(CL);
12180:   SIRegister_TChartBrush(CL);
12181:   TTeeView3DScrolled', Procedure ( IsHoriz : Boolean)
12182:   TTeeView3DChangedZoom', Procedure ( NewZoom : Integer)
12183:   SIRegister_TView3DOptions(CL);
12184:   FindClass('TOBJECT'), 'TTeeCanvas
12185:   TTeeTransparency', 'Integer
12186:   SIRegister_TTeeBlend(CL);
12187:   FindClass('TOBJECT'), 'TCanvas3D
12188:   SIRegister_TTeeShadow(CL);
12189:   teeTGradientDirection', '( gdTopBottom, gdBottomTop, gdLeftRight, g'
12190:   + 'dRightLeft, gdFromCenter, gdFromTopLeft, gdFromBottomLeft, gdRadial, gdDiagonalUp, gdDiagonalDown )
12191:   FindClass('TOBJECT'), 'TSubGradient
12192:   SIRegister_TCustomTeeGradient(CL);
12193:   SIRegister_TSubGradient(CL);
12194:   SIRegister_TTeeGradient(CL);
12195:   SIRegister_TTeeFontGradient(CL);
12196:   SIRegister_TTeeFont(CL);
12197:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12198:   TCanvasTextAlign', 'Integer
12199:   TTeeCanvasHandle', 'HDC
12200:   SIRegister_TTeeCanvas(CL);
12201:   TPoint3DFloat', record X : Double; Y : Double; Z : Double; end
12202:   SIRegister_TFloatXYZ(CL);
12203:   TPoint3D', record x : integer; y : integer; z : Integer; end
12204:   TRGB', record blue: byte; green: byte; red: byte; end
12205:   {TRGB=packed record
12206:     Blue : Byte;
12207:     Green : Byte;
12208:     Red : Byte;
12209:     //$IFDEF CLX //Alpha : Byte; // Linux end;}
12210:
12211:   TTeeCanvasCalcPoints', Function ( x, z : Integer; var P0, P1 : '
12212:     + 'TPoint3D; var Color0, Color1 : TColor) : Boolean

```

```

12213: TTreeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12214: TCanvas3DPlane', '( cpX, cpY, cpZ )
12215: //IInterface', 'IUnknown
12216: SIRegister_TCanvas3D(CL);
12217: SIRegister_TTreeCanvas3D(CL);
12218: TTrianglePoints', 'Array[0..2] of TPoint;
12219: TFourPoints', 'Array[0..3] of TPoint;
12220: Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12221: Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12222: Function Point3D( const x, y, z : Integer) : TPoint3D
12223: Procedure SwapDouble( var a, b : Double)
12224: Procedure SwapInteger( var a, b : Integer)
12225: Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12226: Procedure TeeRectCenter( const R : TRect; var X, Y : Integer)
12227: Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer) : TRect
12228: Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12229: Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12230: Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12231: Procedure UnClipCanvas( ACanvas : TCanvas)
12232: Procedure ClipEllipse( ACanvas : TTreeCanvas; const Rect : TRect)
12233: Procedure ClipRoundRectangle(ACanvas:TTreeCanvas;const Rect : TRect; RoundSize : Integer)
12234: Procedure ClipPolygon(ACanvas:TTreeCanvas;const Points:array of TPoint;NumPoints:Integer)
12235: 'TeeCharForHeight','String 'W
12236: 'DarkerColorQuantity','Byte').SetUInt( 128);
12237: 'DarkColorQuantity','Byte').SetUInt( 64);
12238: TButtonGetColorProc', 'Function : TColor
12239: SIRegister_TTeeButton(CL);
12240: SIRegister_TButtonColor(CL);
12241: SIRegister_TComboFlat(CL);
12242: Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTreeCanvasHandle)
12243: Function TeePoint( const aX, aY : Integer) : TPoint
12244: Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12245: Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12246: Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12247: Function OrientRectangle( const R : TRect) : TRect
12248: Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12249: Function PolygonBounds( const P : array of TPoint) : TRect
12250: Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12251: Function RGBValue( const Color : TColor) : TRGB
12252: Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12253: Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12254: Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12255: Function TeeCull( const P : TFourPoints) : Boolean;
12256: Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12257: TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12258: Procedure SmoothStretch( Src, Dst : TBitmap);
12259: Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12260: Function TeeDistance( const x, y : Double) : Double
12261: Function TeeLoadLibrary( const FileName : String) : HInst
12262: Procedure TeeFreeLibrary( hLibModule : HMODULE)
12263: Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12264: //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12265: Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12266: SIRegister_ICanvasHyperlinks(CL);
12267: SIRegister_ICanvasToolTips(CL);
12268: Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12269: end;
12270:
12271: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12272: begin
12273:   TOvcHdc', 'Integer
12274:   TOvcHWND', 'Cardinal
12275:   TOvcHdc', 'HDC
12276:   TOvcHWND', 'HWND
12277:   Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12278:   Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12279:   Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12280:   Function DefaultEpoch : Integer
12281:   Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12282:   Procedure FixRealPrim( P : PChar; DC : Char)
12283:   Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12284:   Function GetLeftButton : Byte
12285:   Function GetNextDlgItem( Ctrl : TOvcHwnd) : hWnd
12286:   Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12287:   Function GetShiftFlags : Byte
12288:   Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12289:   Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12290:   Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12291:   Function ovIsForegroundTask : Boolean
12292:   Function ovTrimLeft( const S : string) : string
12293:   Function ovTrimRight( const S : string) : string
12294:   Function ovQuotedStr( const S : string) : string
12295:   Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12296:   Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12297:   Function PtrDiff( const P1, P2 : PChar) : Word
12298:   Procedure PtrInc( var P, Delta : Word)
12299:   Procedure PtrDec( var P, Delta : Word)
12300:   Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)

```

```

12301: Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
12302: SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12303: Function ovMinI( X, Y : Integer) : Integer
12304: Function ovMaxI( X, Y : Integer) : Integer
12305: Function ovMinL( X, Y : LongInt) : LongInt
12306: Function ovMaxL( X, Y : LongInt) : LongInt
12307: Function GenerateComponentName( PF : TWInControl; const Root : string) : string
12308: Function PartialCompare( const S1, S2 : string) : Boolean
12309: Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12310: Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12311: Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12312: Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
TransparentColor : TColor)
12313: Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
TransparentColor : TColorRef)
12314: Function ovWidthOf( const R : TRect) : Integer
12315: Function ovHeightOf( const R : TRect) : Integer
12316: Procedure ovDebugOutput( const S : string)
12317: Function GetArrowWidth( Width, Height : Integer) : Integer
12318: Procedure StripCharSeq( CharSeq : string; var Str : string)
12319: Procedure StripCharFromEnd( aChr : Char; var Str : string)
12320: Procedure StripCharFromFront( aChr : Char; var Str : string)
12321: Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12322: Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12323: Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12324: Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12325: Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12326: Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12327: Function CreateMetaFile( p1 : PChar) : HDC
12328: Function DescribePixelFormat(DC: HDC;p2:INT;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12329: Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12330: Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12331: Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12332: Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12333: Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12334: Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12335: //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12336: Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12337: Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12338: //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12339: Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12340: Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
SrcHeight:Int;Rop:DWORD):BOOL
12341: Function SetRectRgn( Rgn : HRGN; X1, Y1, X2, Y2 : Integer) : BOOL
12342: Function StretchDIBits(DC : HDC; DestX, DestY, DestWidth, DestHeight, SrcX, SrcY, SrcWidth,
SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12343: Function SetROP2( DC : HDC; p2 : Integer) : Integer
12344: Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12345: Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12346: Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12347: Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12348: Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12349: Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12350: Function UpdateColors( DC : HDC) : BOOL
12351: Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12352: Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12353: Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12354: Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12355: Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12356: Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12357: Function MaskBlt( DestDC : HDC; XDest, YDest, Width, Height : Integer; SrcDC : HDC; XSrc, YSrc : Integer;
Mask : HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12358: Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
yMask:Int):BOOL;
12359: Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12360: Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12361: Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12362: Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12363: Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12364: Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12365: Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12366: Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12367: Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12368: Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12369: Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12370: Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12371: end;
12372: procedure SIRegister_ovcfile(CL: TPSPascalCompiler);
12373: begin
12374:   SIRegister_TOvcAbstractStore(CL);
12375:   //PEXPropInfo', '^TexPropInfo // will not work
12376:   // TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12377:   SIRegister_TOvcPropertyList(CL);
12378:   SIRegister_TOvcDataFiler(CL);
12379:   Procedure UpdateStoredList( AForm : TWInControl; AStoredList : TStrings; FromForm : Boolean)
12380:   Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12381:   Function CreateStoredItem( const CompName, PropName : string) : string
12382:   Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean

```

```

12383: //Function GetPropType( PropInfo : PExPropInfo ) : PTypeInfo
12384: end;
12385:
12386: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12387: begin
12388:   'ovsetsize','LongInt'( 16);
12389:   'etSyntax','LongInt'( 0);
12390:   'etSymantic','LongInt'( 1);
12391:   'chCR','Char #13);
12392:   'chLF','Char #10);
12393:   'chLineSeparator',' chCR);
12394:   SIRegister_TCocoError(CL);
12395:   SIRegister_TCommentItem(CL);
12396:   SIRegister_TCommentList(CL);
12397:   SIRegister_TSymbolPosition(CL);
12398: TGenListType', '( glNever, glAlways, glOnError )
12399: TovBitSet', 'set of Integer
12400: //PStartTable', '^TStartTable // will not work
12401: 'TovCharSet', 'set of AnsiChar
12402: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12403: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12404: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12405: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12406: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12407:   +osition; const Data : string; ErrorType : integer)
12408: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12409: TGetCH', 'Function ( pos : longint) : char
12410: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12411:   SIRegister_TCocoRScanner(CL);
12412:   SIRegister_TCocoRGrammar(CL);
12413:   '_EF','Char #0);
12414:   '_TAB','Char').SetString( #09);
12415:   '_CR','Char').SetString( #13);
12416:   '_LF','Char').SetString( #10);
12417:   '_EL','').SetString( _CR);
12418:   '_EOF','Char').SetString( #26);
12419:   'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12420:   'minErrDist','LongInt'( 2);
12421:   Function ovPadL( S : string; ch : char; L : integer) : string
12422: end;
12423:
12424:   TFormatSettings = record
12425:     CurrencyFormat: Byte;
12426:     NegCurrFormat: Byte;
12427:     ThousandSeparator: Char;
12428:     DecimalSeparator: Char;
12429:     CurrencyDecimals: Byte;
12430:     DateSeparator: Char;
12431:     TimeSeparator: Char;
12432:     ListSeparator: Char;
12433:     CurrencyString: string;
12434:     ShortDateFormat: string;
12435:     LongDateFormat: string;
12436:     TimeAMString: string;
12437:     TimePMString: string;
12438:     ShortTimeFormat: string;
12439:     LongTimeFormat: string;
12440:     ShortMonthNames: array[1..12] of string;
12441:     LongMonthNames: array[1..12] of string;
12442:     ShortDayNames: array[1..7] of string;
12443:     LongDayNames: array[1..7] of string;
12444:     TwoDigitYearCenturyWindow: Word;
12445:   end;
12446:
12447: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12448: begin
12449:   Function ovFormatSettings : TFormatSettings
12450: end;
12451:
12452: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12453: begin
12454:   'TovcCharSet', 'set of Char
12455:   ovBTable', 'array[0..255] of Byte
12456:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
12457:   Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12458:   Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12459:   Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12460:   Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12461:   Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12462:   Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12463:   Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12464:   Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12465:   Function HexBPChar( Dest : PChar; B : Byte) : PChar
12466:   Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12467:   Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12468:   Function HexWPChar( Dest : PChar; W : Word) : PChar
12469:   Function LoCaseChar( C : Char) : Char
12470:   Function OctalLPChar( Dest : PChar; L : LongInt) : PChar

```



```

12471: Function StrChDeletePrim( P : PChar; Pos : Cardinal ) : PChar
12472: Function StrChInsertPrim( Dest : PChar; C : Char; Pos : Cardinal ) : PChar
12473: Function StrChPos( P : PChar; C : Char; var Pos : Cardinal ) : Boolean
12474: Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word )
12475: Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal ) : PChar
12476: Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal ) : PChar
12477: Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal ) : PChar
12478: Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal ) : PChar
12479: Function StrStPos( P, S : PChar; var Pos : Cardinal ) : Boolean
12480: Function StrToLongPChar( S : PChar; var I : LongInt ) : Boolean
12481: Procedure TrimAllSpacesPChar( P : PChar )
12482: Function TrimEmbeddedZeros( const S : string ) : string
12483: Procedure TrimEmbeddedZerosPChar( P : PChar )
12484: Function TrimTrailPrimPChar( S : PChar ) : PChar
12485: Function TrimTrailPChar( Dest, S : PChar ) : PChar
12486: Function TrimTrailingZeros( const S : string ) : string
12487: Procedure TrimTrailingZerosPChar( P : PChar )
12488: Function UpCaseChar( C : Char ) : Char
12489: Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet ) : Boolean
12490: Function ovc32StringIsCurrentCodePage( const S : WideString ) : Boolean;
12491: //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal ) : Boolean;
12492: end;
12493:
12494: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12495: begin
12496:   //PraiseFrame', '^TRaiseFrame // will not work
12497:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : __Poin'
12498:   + 'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12499:   Procedure SafeCloseHandle( var Handle : THandle )
12500:   Procedure ExchangeInteger( X1, X2 : Integer )
12501:   Procedure FillInteger( const Buffer, Size, Value : Integer )
12502:   Function LongMulDiv( Mult1, Mult2, Div1 : Longint ) : Longint
12503:   Function afCompareMem( P1, P2 : TObject; Length : Integer ) : Boolean
12504:
12505: FILENAME_ADVAPI32 = 'ADVAPI32.DLL';
12506:   function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12507: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12508:   function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12509:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12510:   SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12511:   const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12512:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12513:   function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12514:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12515:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12516:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12517:   ObjectTypeInfoLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12518:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12519:   function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12520:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12521:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12522:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12523:   ObjectTypeInfoLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12524:   var GrantedAccess: DWORD; var AccessStatusList: DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12525:   function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12526:   function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12527:   function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12528:   lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12529:   lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12530:   dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12531:   const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12532:   function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12533:   function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12534:   pSecurityDescriptor: PSecurityDescriptor; nLength: DWORD; var lpnLengthNeeded: DWORD): BOOL; stdcall;
12535:   function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12536:   function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12537:   dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12538:   function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12539:   dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12540:   function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12541:   Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12542:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12543:   function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12544:   Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12545:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12546:   function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12547:   lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12548:   function LookupPrivilegeName(lpSystemName: PKOLChar;
12549:   var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12550:   function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12551:   var lpLuid: TLargeInteger): BOOL; stdcall;
12552:   function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12553:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12554:   function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12555:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12556:   function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12557:   ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12558:   ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12559:   var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;

```

```

12560:     var GenerateOnClose: BOOL): BOOL; stdcall;
12561: function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12562:   HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12563:   var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12564: function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12565: function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12566: function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12567:   ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12568: function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12569:   lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12570:   var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12571: function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12572:   var phkResult: HKEY): Longint; stdcall;
12573: function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12574:   var phkResult: HKEY): Longint; stdcall;
12575: function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12576:   Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12577:   lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12578:   lpDisposition: PDWORD): Longint; stdcall;
12579: function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12580: function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12581: function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12582:   var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12583:   lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12584: function RegEnumKey(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar; cbName: DWORD): Longint; stdcall;
12585: function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12586:   var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12587:   lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12588: function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12589: function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar; var phkResult: HKEY): Longint; stdcall;
12590: function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12591:   ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12592: function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12593:   lpcbClass: PDWORD; lpReserved: Pointer;
12594:   lpcbSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcbValues,
12595:   lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12596:   lpftLastWriteTime: PFileTime): Longint; stdcall;
12597: function RegQueryMultipleValues(hKey: HKEY; var Vallist;
12598:   NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotSize: DWORD): Longint; stdcall;
12599: function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12600:   lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12601: function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12602:   lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12603: function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12604:   lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12605: function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12606: function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12607:   lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12608: function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12609:   dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12610: function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12611:   Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12612: function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12613: function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12614: function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12615:   dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12616:   dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12617: function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12618:   pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12619:
12620: Function wAddAtom(lpString: PKOLChar): ATOM
12621: Function wBeginUpdateResource(pFileName: PKOLChar; bDeleteExistingResources: BOOL): THandle
12622: //Function wCallNamedPipe(lpNamedPipeName: PKOLChar; lpInBuffer: Pointer; nInBufferSize: DWORD;
12623:   lpOutBuffer: Pointer; nOutBufferSize: DWORD; var lpBytesRead: DWORD; nTimeOut: DWORD): BOOL
12624: Function wCommConfigDialog(lpszName: PKOLChar; hWnd: HWND; var lpCC: TCommConfig): BOOL
12625: Function wCompareString(Locale: LCID; dwCmpFlags: DWORD; lpString1: PKOLChar; cchCount1: Integer;
12626:   lpString2: PKOLChar; cchCount2: Integer): Integer
12627: Function wCopyFile(lpExistingFileName, lpNewFileName: PKOLChar; bFailIfExists: BOOL): BOOL
12628: //Function wCopyFileEx(lpExistingFileName, lpNewFileName: PKOLChar; lpProgressRoutine:
12629:   TFPNProgressRoutine; lpData: Pointer; pbCancel: PBool; dwCopyFlags: DWORD): BOOL
12630: Function wCreateDirectory(lpPathName: PKOLChar; lpSecurityAttributes: PSecurityAttributes): BOOL
12631: Function wCreateDirectoryEx(lpTemplateDirectory,
12632:   lpNewDirectory: PKOLChar; lpSecAttrib: PSecurityAttributes): BOOL;
12633: Function wCreateEvent(lpEventAttributes: PSecurityAttributes; bManualReset,
12634:   bInitialState: BOOL; lpName: PKOLChar): THandle;
12635: Function wCreateFile(lpFileName: PKOLChar; dwDesiredAccess, dwShareMode: DWORD; lpSecurityAttributes:
12636:   PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes: DWORD; hTemplateFile: THandle): THandle
12637: Function wCreateFileMapping(hFile: THandle; lpFileMappingAttributes: PSecurityAttributes; flProtect,
12638:   dwMaximumSizeHigh, dwMaximumSizeLow: DWORD; lpName: PKOLChar): THandle
12639: Function wCreateHardLink(lpFileName,
12640:   lpExistingFileName: PKOLChar; lpSecurityAttributes: PSecurityAttributes): BOOL
12641: Function
12642: CreateMailslot(lpName: PKOLChar; MaxMessSize: DWORD; lReadTimeout: DWORD; lpSecurityAttrib: PSecurityAttributes): THandle;
12643: Function wCreateNamedPipe(lpName: PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
12644:   nInBufferSize, nDefaultTimeOut: DWORD; lpSecurityAttributes: PSecurityAttributes): THandle
12645: //Function CreateProcess(lpApplicationName: PKOLChar; lpCommandLine: PKOLChar; lpProcessAttributes,
12646:   lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL; dwCreationFlags: DWORD; lpEnvironment:
12647:   Pointer; lpCurrentDirectory: PKOLChar; const lpStartupInfo: TStartupInfo; var
12648:   lpProcessInfo: TProcessInformation): BOOL

```

```

12636: Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
Longint; lpName : PKOLChar) : THandle
12637: Function
wCreateWaitableTimer(lpTimerAttributes:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12638: Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12639: Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12640: Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12641: //Function
wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12642: //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12643: //Function
wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12644: //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12645: //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12646: //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12647: //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12648: Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12649: Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12650: //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12651: Function wFindAtom( lpString : PKOLChar) : ATOM
12652: Function
wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12653: Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12654: //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindInfoLevels; lpFindFileData :
Pointer; fSearchOp : TFindIndexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12655: Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12656: Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12657: Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12658: Function
wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12659: //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12660: Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12661: Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12662: Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12663: Function wGetCommandLine : PKOLChar
12664: //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12665: Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12666: Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12667: //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12668: Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12669: //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
lpDateStr : PKOLChar; cchDate : Integer) : Integer
12670: //Function wGetDefaultCommConfig( lpSzName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12671: Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12672: //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12673: Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12674: Function wGetEnvironmentStrings : PKOLChar
12675: Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12676: Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12677: //Function
wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12678: Function wGetFullPathName(lpFileName:PKOLChar;nBufferLength:WORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12679: //Function wGetLocaleInfo(Locale:LCID; LType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12680: Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12681: Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12682: Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12683: //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12684: //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12685: Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12686: Function
wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12687: Function wGetPrivateProfileSectionNames(lpSzReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12688: Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr : PKOLChar;
nSize:DWORD; lpFileName : PKOLChar) : DWORD
12689: Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12690: Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12691: Function wGetProfileString(lpAppName,lpKeyName,
lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12692: Function wGetShortPathName( lpSzLongPath:PKOLChar;lpSzShortPath : PKOLChar; cchBuffer : DWORD) : DWORD
12693: //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12694: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
lpCharType):BOOL
12695: Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12696: Function wGetTempFileName( lpPathName, lpPrefixString : PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12697: Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12698: //Function
wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12699: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12700: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize :
DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL

```



```

12701: Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT ) : UINT
12702: Function wGlobalAddAtom( lpString : PKOLChar ) : ATOM
12703: Function wGlobalFindAtom( lpString : PKOLChar ) : ATOM
12704: Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer ) : UINT
12705: Function wisBadStringPtr( lpsz : PKOLChar; ucchMax : UINT ) : BOOL
12706: Function
wLCMapString( Loc:LCID; dwMapFlgs:DWORD; lpSrcStr:PKOLChar; cchSrc:Int; lpDestStr:PKOLChar; cchDest:Int):Int;
12707: Function wLoadLibrary( lpLibFileName : PKOLChar ) : HMODULE
12708: Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD ) : HMODULE
12709: Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar ) : BOOL
12710: Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD ) : BOOL
12711: //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD ) : BOOL
12712: Function wOpenEvent( dwDesiredAccess : DWORD; binheritHandle : BOOL; lpName:PKOLChar ) : THandle
12713: Function wOpenFileMapping( dwDesiredAccess : DWORD; binheritHandle:BOOL; lpName: PKOLChar):THandle
12714: Function wOpenMutex( dwDesiredAccess : DWORD; binheritHandle : BOOL; lpName : PKOLChar ) : THandle
12715: Function wOpenSemaphore( dwDesiredAccess : DWORD; binheritHandle : BOOL; lpName : PKOLChar):THandle
12716: Function wOpenWaitableTimer(dwDesiredAccess:DWORD;binheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12717: Procedure wOutputDebugString( lpOutputString : PKOLChar)
12718: //Function wPeekConsoleInput( hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
lpNumberOfEventsRead:DWORD):BOOL;
12719: Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD ) : DWORD
12720: //Function wQueryRecoveryAgents( pl:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12721: //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
lpNumberOfCharsRead : DWORD; lpReserved : Pointer ) : BOOL
12722: //Function wReadConsoleInput( hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
lpNumOfEventsRead:DWORD):BOOL;
12723: //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
: TCoord; var lpReadRegion : TSmallRect ) : BOOL
12724: //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD ) : BOOL
12725: Function wRemoveDirectory( lpPathName : PKOLChar ) : BOOL
12726: //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo ) : BOOL
12727: Function wSearchPath( lpPath, lpFileName, lpExtension:PKOLChar; nBufferLength:DWORD; lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12728: Function wSetComputerName( lpComputerName : PKOLChar ) : BOOL
12729: Function wSetConsoleTitle( lpConsoleTitle : PKOLChar ) : BOOL
12730: Function wSetCurrentDirectory( lpPathName : PKOLChar ) : BOOL
12731: //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD ) : BOOL
12732: Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar ) : BOOL
12733: Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD ) : BOOL
12734: //Function wSetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCData : PKOLChar ) : BOOL
12735: Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar ) : BOOL
12736: //Function wUpdateResource( hUpdate:THandle; lpType,
lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12737: Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD ) : DWORD
12738: Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeout : DWORD ) : BOOL
12739: //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer ) : BOOL
12740: //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
var lpNumberOfEventsWritten : DWORD ) : BOOL
12741: //Function wWriteConsoleOutput( hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
TCoord; var lpWriteRegion : TSmallRect ) : BOOL
12742: //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12743: Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar ) : BOOL
12744: Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar ) : BOOL
12745: Function wWriteProfileSection( lpAppName, lpString : PKOLChar ) : BOOL
12746: Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar ) : BOOL
12747: Function wlstcat( lpString1, lpString2 : PKOLChar ) : PKOLChar
12748: Function wlstcomp( lpString1, lpString2 : PKOLChar ) : Integer
12749: Function wlstcmpi( lpString1, lpString2 : PKOLChar ) : Integer
12750: Function wlstcopy( lpString1, lpString2 : PKOLChar ) : PKOLChar
12751: Function wlstcncpy( lpString1, lpString2 : PKOLChar; iMaxLength : Integer ) : PKOLChar
12752: Function wlstrlen( lpString : PKOLChar ) : Integer
12753: //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
PNetConnectInfoStruct ) : DWORD
12754: //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassword,
lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12755: //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
lpUserName:PKOLChar; dwFlags : DWORD ) : DWORD
12756: Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar ) : DWORD
12757: Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL ) : DWORD
12758: Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL ) : DWORD
12759: //Function wWNetConnectDialog1( var lpConnDlgStruct : TConnectDlgStruct ) : DWORD
12760: //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct ) : DWORD
12761: //Function wWNetEnumResource( hEnum:THandle;var lpCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12762: Function wWNetGetConnection( lpLocalName:PKOLChar; lpRemoteName:PKOLChar; var lpnLength:DWORD ) : DWORD;
12763: Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
: PKOLChar; nNameBufSize : DWORD ) : DWORD
12764: //Function wWNetGetNetworkInformation( lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12765: Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12766: //Function wWNetGetResourceParent( lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12767: //Function wWNetGetUniversalName( lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
lpBufferSize:DWORD):DWORD;
12768: Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD ) : DWORD
12769: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lpEnum:THandle):DWORD;
12770: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer ) : DWORD

```



```

12771: //Function wWNetUseConnection(hwndOwner:HWND;var
lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12772: Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12773: Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12774: Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12775: Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12776: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lpBuffer:Ptr;var puLen:UINT):BOOL;
12777: //Function wGetPrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12778: //Function wWritePrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12779: Function wAddFontResource( FileName : PKOLChar) : Integer
12780: //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12781: Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12782: Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12783: //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12784: //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12785: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12786: Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
fdwUnderline, fdwStrikeOut, fdwCharSet, fdwOutputPrec, fdwClipPrecision, fdwQualy,
fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12787: Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12788: //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12789: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12790: Function wCreateMetaFile( p1 : PKOLChar) : HDC
12791: Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12792: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12793: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFontEnumProc; p4 : LPARAM) : BOOL
12794: //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12795: //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntemprc:TFontEnumProc;lpszData:PKOLChar):Integer;
12796: //Function wEnumICMProfiles( DC : HDC; ICMProc : TFontICMEnumProc; p3 : LPARAM) : Integer
12797: //Function wExtTextOut(DC:HDC;X,
Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12798: //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12799: //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatStructs) : BOOL
12800: //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12801: //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12802: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12803: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPRResults;p6:DWORD):DWORD
12804: Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12805: Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12806: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12807: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12808: Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12809: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12810: Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12811: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12812: //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12813: //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12814: Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12815: Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12816: Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12817: //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12818: Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12819: Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12820: //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12821: //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12822: Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12823: //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12824: Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12825: Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12826: Function wWglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12827: //Function wWglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12828: Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12829: Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12830: //Function
wCallWindowProc( lpPrevWndFunc:TFontWndProc;hwnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12831: //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12832: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
dwFlags : DWORD; lParam : Pointer) : Longint
12833: Function wChangeMenu( hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12834: Function wCharLower( lpsz : PKOLChar) : PKOLChar
12835: Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12836: Function wCharNext( lpsz : PKOLChar) : PKOLChar
12837: //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12838: Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12839: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12840: Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12841: Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12842: Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12843: Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12844: Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12845: Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12846: //Function wCreateDesktop(lpszDesktop,
lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK

```

```

12847: //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
      HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND
12848: //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
      lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND
12849: Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
      hWndParent : HWND; hInstance : HINST; lParam : LPARAM ) : HWND
12850: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
      nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12851: //Function wCreateWindowStation(lpinsta:PKOLChar;dwReserv,
      dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12852: Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12853: Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12854: Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT;
12855: Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12856: //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
      : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12857: //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
      : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12858: Function wDispatchMessage( const lpMsg : TMsg ) : Longint
12859: Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
      nIDStaticPath:Integer;uFileType:UINT):Integer;
12860: Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
      nIDStaticPath:Int;uFileType:UINT):Int;
12861: Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer) : BOOL
12862: Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer ) : BOOL
12863: //Function wDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
      cy:Int;Flags:UINT):BOOL;
12864: Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12865: Function wFindWindow( lpClassName, lpWindowName : PKOLChar ) : HWND
12866: Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar ) : HWND
12867: //Function wGetAltTabInfo(hWnd:HWND;iItem:Int;var
      pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12868: //Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass ) : BOOL
12869: //Function wGetClassInfoEx( Instance : HINST; ClassName : PKOLChar; var WndClass : TWndClassEx ) : BOOL
12870: Function wGetClassLong( hWnd : HWND; nIndex : Integer ) : DWORD
12871: Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer ) : Integer
12872: Function wGetClipboardFormatName( format : UINT; lpzFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12873: Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12874: Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer ) : Integer
12875: Function wGetKeyboardLayoutName( pwszKLID : PKOLChar ) : BOOL
12876: //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo ) : BOOL
12877: Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12878: Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wParam:FilterMin, wParam:FilterMax : UINT ) : BOOL
12879: Function wGetProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12880: //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
      lpnTabStopPositions ) : DWORD
12881: //Function wGetObjectInformation(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
      lpnLengthNeed:DWORD)BOOL;
12882: Function wGetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
12883: Function wGetWindowModuleFileName( hWnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT ) : UINT
12884: Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer ) : Integer
12885: Function wGetWindowTextLength( hWnd : HWND ) : Integer
12886: //Function wGrayString( hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
      nHeigt:Int):BOOL;
12887: Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12888: //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
12889: Function wIsCharAlpha( ch : KOLChar ) : BOOL
12890: Function wIsCharAlphaNumeric( ch : KOLChar ) : BOOL
12891: Function wIsCharLower( ch : KOLChar ) : BOOL
12892: Function wIsCharUpper( ch : KOLChar ) : BOOL
12893: Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg ) : BOOL
12894: Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar ) : HACCEL
12895: Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar ) : HBITMAP
12896: Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar ) : HCURSOR
12897: Function wLoadCursorFromFile( lpFileName : PKOLChar ) : HCURSOR
12898: Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar ) : HICON
12899: Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12900: Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT ) : HKL
12901: Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar ) : HMENU
12902: //Function wLoadMenuIndirect( lpMenuTemplate : Pointer ) : HMENU
12903: Function wLoadString(hInstance:HINST;uID:UINT;lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12904: Function wMapVirtualKey( uCode, uMapType : UINT ) : UINT
12905: Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwkl : HKL ) : UINT
12906: Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT ) : Integer
12907: Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12908: //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams ) : BOOL
12909: Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12910: //Function wOemToAnsi( const lpzSrc : LPCSTR; lpzDst : LPSTR ) : BOOL
12911: //7Function wOemToAnsiBuff( lpzSrc : LPCSTR; lpzDst : LPSTR; cchDstLength : DWORD ) : BOOL
12912: //Function wOemToChar( lpzSrc : PKOLChar; lpzDst : PKOLChar ) : BOOL
12913: Function wOemToCharBuff( lpzSrc : PKOLChar; lpzDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12914: Function wOpenDesktop(lpzDesktop:PKOLChar;dwFlags:DWORD;finherit:BOOL;dwDesiredAccess:DWORD): HDESK
12915: Function wOpenWindowStation( lpzWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD ) : HWINSTA
12916: Function wPeekMessage( var lpMsg : TMsg; hWnd : HWND; wParam:FilterMin,wParam:FilterMax, wRemoveMsg:UINT):BOOL
12917: Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12918: Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12919: Function wRealGetWindowClass( hWnd : HWND; pszType : PKOLChar; cchType : UINT ) : UINT
12920: //Function wRegisterClass( const lpWndClass : TWndClass ) : ATOM
12921: //Function wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM

```

```

12922: Function wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
12923: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12924: Function wRegisterWindowMessage( lpString : PKOLChar ) : UINT
12925: Function wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12926: Function wSendDlgiItemMessage(hDlgi:HWND;nIDDLgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12927: Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12928: //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
lpResultCallback : TFNSendAsyncProc; dwData : DWORD ) : BOOL
12929: Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
lpdwResult:DWORD): LRESULT
12930: Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12931: Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
12932: Function wSetDlgItemText( hDlg : HWND; nIDDLgItem : Integer; lpString : PKOLChar ) : BOOL
12933: //Function wSetMenuItemInfo( p1 : HMENU; p2 : Integer; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
12934: Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle ) : BOOL
12935: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
12936: Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint
12937: Function wSetWindowText( hWnd : HWND; lpString : PKOLChar ) : BOOL
12938: //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc ) : HHOOK
12939: //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
12940: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
12941: Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
lpnTabStopPositions,nTabOrigin:Int):Longint;
12942: Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg ) : Integer
12943: Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST ) : BOOL
12944: Function wVkKeyScan( ch : KOLChar ) : SHORT
12945: Function wVkKeyScanEx( ch : KOLChar; dwHKL : HKL ) : SHORT
12946: Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD ) : BOOL
12947: Function wwsprintf( Output : PKOLChar; Format : PKOLChar ) : Integer
12948: Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list ) : Integer
12949:
12950: //TestDrive!
12951: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
12952: 'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString('ConvertSidToStringSida
12953: Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
12954: Function GetLocalUserSidStr( const UserName : string ) : string
12955: Function getPid4user( const domain : string; const user : string; var pid : dword ) : boolean
12956: Function Impersonate2User( const domain : string; const user : string ) : boolean
12957: Function GetProcessUserByPid( pid : DWORD; var UserName, Domain : AnsiString ) : Boolean
12958: Function KillProcessbyname( const exename : string; var found : integer ) : integer
12959: Function getWinProcessList : TStringList
12960: end;
12961:
12962: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
12963: begin
12964: 'AfMaxSyncSlots','LongInt'( 64);
12965: 'AfSynchronizeTimeout','LongInt'( 2000);
12966: TafSyncSlotID', 'DWORD
12967: TafSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
12968: TafSafeSyncEvent', 'Procedure ( ID : TafSyncSlotID)
12969: TafSafeDirectSyncEvent', 'Procedure
12970: Function AfNewSyncSlot( const AEvent : TafSafeSyncEvent ) : TafSyncSlotID
12971: Function AfReleaseSyncSlot( const ID : TafSyncSlotID ) : Boolean
12972: Function AfEnableSyncSlot( const ID : TafSyncSlotID; Enable : Boolean ) : Boolean
12973: Function AfValidateSyncSlot( const ID : TafSyncSlotID ) : Boolean
12974: Function AfSyncEvent( const ID : TafSyncSlotID; Timeout : DWORD ) : Boolean
12975: Function AfDirectSyncEvent( Event : TafSafeDirectSyncEvent; Timeout : DWORD ) : Boolean
12976: Function AfIsSyncMethod : Boolean
12977: Function AfSyncWnd : HWND
12978: Function AfSyncStatistics : TafSyncStatistics
12979: Procedure AfClearSyncStatistics
12980: end;
12981:
12982: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
12983: begin
12984: 'fBinary','LongWord'($00000001);
12985: 'fParity','LongWord'($00000002);
12986: 'fOutxCtsFlow','LongWord').SetUInt($00000004);
12987: 'fOutxDsrFlow','LongWord'($00000008);
12988: 'fDtrControl','LongWord'($00000030);
12989: 'fDtrControlDisable','LongWord'($00000000);
12990: 'fDtrControlEnable','LongWord'($00000010);
12991: 'fDtrControlHandshake','LongWord'($00000020);
12992: 'fDsrSensitivity','LongWord'($00000040);
12993: 'fTXContinueOnXoff','LongWord'($00000080);
12994: 'fOutX','LongWord'($00000100);
12995: 'fInX','LongWord'($00000200);
12996: 'fErrorChar','LongWord'($00000400);
12997: 'fNull','LongWord'($00000800);
12998: 'fRtsControl','LongWord'($00003000);
12999: 'fRtsControlDisable','LongWord'($00000000);
13000: 'fRtsControlEnable','LongWord'($00001000);
13001: 'fRtsControlHandshake','LongWord'($00002000);
13002: 'fRtsControlToggle','LongWord'($00003000);
13003: 'fAbortOnError','LongWord'($00004000);
13004: 'fDummy2','LongWord'($FFFF8000);
13005: TafCoreEvent', ( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13006: FindClass('TOBJECT'),'EafComPortCoreError
13007: FindClass('TOBJECT'),'TafComPortCore

```

```

13008:   TAFComPortCoreEvent', 'Procedure ( Sender : TAFComPortCore; Even'
13009:     + 'tKind : TAFCoreEvent; Data : DWORD)
13010:   SIRegister_TAFComPortCoreThread(CL);
13011:   SIRegister_TAFComPortEventThread(CL);
13012:   SIRegister_TAFComPortWriteThread(CL);
13013:   SIRegister_TAFComPortCore(CL);
13014:   Function FormatDeviceName( PortNumber : Integer) : string
13015: end;
13016:
13017: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13018: begin
13019:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13020:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13021:   SIRegister_TApplicationFileIO(CL);
13022:   TDataFileCapability', '( dfcRead, dfcWrite )
13023:   TDataFileCapabilities', 'set of TDataFileCapability
13024:   SIRegister_TDataFile(CL);
13025:   //TDataFileClass', 'class of TDataFile
13026:   Function ApplicationFileIODefined : Boolean
13027:   Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13028:   Function FileStreamExists(const fileName: String) : Boolean
13029:   //Procedure Register
13030: end;
13031:
13032: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13033: begin
13034:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13035:   + ', uftCString, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13036:   + 'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13037:   TALFBXScale', 'Integer
13038:   FindClass('TOBJECT'), 'EALFBXConvertError
13039:   SIRegister_EALFBXError(CL);
13040:   SIRegister_EALFBXException(CL);
13041:   FindClass('TOBJECT'), 'EALFBXGFixError
13042:   FindClass('TOBJECT'), 'EALFBXDSQLError
13043:   FindClass('TOBJECT'), 'EALFBXDynError
13044:   FindClass('TOBJECT'), 'EALFBXGBakError
13045:   FindClass('TOBJECT'), 'EALFBXGSecError
13046:   FindClass('TOBJECT'), 'EALFBXLICENSEError
13047:   FindClass('TOBJECT'), 'EALFBXGStatError
13048:   //EALFBXExceptionClass', 'class of EALFBXError
13049:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13050:   + '37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13051:   + 'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13052:   + 'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252, '
13053:   + 'csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13054:   + 'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13055:   + '_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13056:   + '8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13057:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13058:   + 'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13059:   + 'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13060:   + 'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13061:   TALFBXTransParams', 'set of TALFBXTransParam
13062:   Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13063:   Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13064:   Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13065:   'cALFBXMaxParamLength', 'LongInt'( 125);
13066:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13067:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13068:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13069:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13070:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete, '
13071:   + 'stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stComm'
13072:   + 'it, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13073:   SIRegister_TALFBXSQLDA(CL);
13074:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13075:   SIRegister_TALFBXPoolStream(CL);
13076:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13077:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13078:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13079:   //TALFBXArrayDesc', 'TISCArrDesc
13080:   //TALFBXBlobDesc', 'TISCBlobDesc
13081:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13082:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13083:   SIRegister_TALFBXSQLResult(CL);
13084:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13085:   SIRegister_TALFBXSQLParams(CL);
13086:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13087:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13088:   + 'atementType : TALFBXStatementType; end
13089:   FindClass('TOBJECT'), 'TALFBXLibrary
13090:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13091:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13092:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13093:   // '+' Excep : EALFBXExceptionClass)
13094:   SIRegister_TALFBXLibrary(CL);
13095:   'cALFBXDateOffset', 'LongInt'( 15018);
13096:   'cALFBXTimeCoeff', 'LongInt'( 86400000);

```



```

13097: //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13098: //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13099: //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13100: Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13101: Procedure ALFBXDecodeSQLTime( v : Cardinal; out Hour, Minute, Second : Word; out Fractions : LongWord)
13102: //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13103: //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13104: //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13105: Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13106: Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord) : Cardinal
13107: TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13108: TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13109: Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13110: Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13111: end;
13112:
13113: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13114: begin
13115:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13116:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13117:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13118:   + 'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13119:   + 'eger; First : Integer; CacheThreshold : Integer; end
13120:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13121:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params : TALFBXClientSQLParams; end
13122:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13123:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13124:   + '_writes : int64; page_fetches : int64; page_marks : int64; end
13125:   SIRegister_TALFBXClient(CL);
13126:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13127:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13128:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13129:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13130:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13131:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13132:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13133:   SIRegister_TALFBXConnectionPoolClient(CL);
13134:   SIRegister_TALFBXEventThread(CL);
13135:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13136: end;
13137:
13138: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13139: begin
13140:   _OSVERSIONINFOA = record
13141:     dwOSVersionInfoSize: DWORD;
13142:     dwMajorVersion: DWORD;
13143:     dwMinorVersion: DWORD;
13144:     dwBuildNumber: DWORD;
13145:     dwPlatformId: DWORD;
13146:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13147:   end;
13148:   TOSVersionInfoA', '_OSVERSIONINFOA
13149:   TOSVersionInfo', 'TOSVersionInfoA
13150:   'WS_EX_RIGHT', 'LongWord')( $00001000);
13151:   'WS_EX_LEFT', 'LongWord')( $00000000);
13152:   'WS_EX_RTLREADING', 'LongWord')( $00002000);
13153:   'WS_EX_LTRREADING', 'LongWord')( $00000000);
13154:   'WS_EX_LEFTSCROLLBAR', 'LongWord')( $00004000);
13155:   'WS_EX_RIGHTSCROLLBAR', 'LongWord')( $00000000);
13156:   Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13157:   'LAYOUT_RTL', 'LongWord')( $00000001);
13158:   'LAYOUT_BTT', 'LongWord')( $00000002);
13159:   'LAYOUT_VBH', 'LongWord')( $00000004);
13160:   'LAYOUT_BITMAPORIENTATIONPRESERVED', 'LongWord')( $00000008);
13161:   'NOMIRRORBITMAP', 'LongWord')( DWORD ( $80000000 ));
13162:   Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13163:   Function GetLayout( dc : hdc) : DWORD
13164:   Function IsBidi : Boolean
13165:   Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13166:   Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13167:   Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13168:   Function GetPriorityClass( hProcess : THandle) : DWORD
13169:   Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13170:   Function CloseClipboard : BOOL
13171:   Function GetClipboardSequenceNumber : DWORD
13172:   Function GetClipboardOwner : HWND
13173:   Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13174:   Function GetClipboardViewer : HWND
13175:   Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13176:   Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13177:   Function GetClipboardData( uFormat : UINT) : THandle
13178:   Function RegisterClipboardFormat( lpzFormat : PChar) : UINT
13179:   Function CountClipboardFormats : Integer
13180:   Function EnumClipboardFormats( format : UINT) : UINT
13181:   Function GetClipboardFormatName( format:UINT;lpzFormatName:PChar;cchMaxCount:Integer):Integer
13182:   Function EmptyClipboard : BOOL
13183:   Function IsClipboardFormatAvailable( format : UINT) : BOOL
13184:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13185:   Function GetOpenClipboardWindow : HWND

```

```

13186: Function EndDialog( hDlg : HWND; nResult : Integer ) : BOOL
13187: Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer ) : HWND
13188: Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL ): BOOL
13189: Function GetDlgItemInt( hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL ): UINT
13190: Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar ) : BOOL
13191: Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT ) : BOOL
13192: Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer ) : BOOL
13193: Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer ) : UINT
13194: Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13195: end;
13196:
13197: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13198: begin
13199:   Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13200:   Function GetTemporaryFilesPath : String
13201:   Function GetTemporaryFileName : String
13202:   Function FindFileInPaths( const fileName, paths : String ) : String
13203:   Function PathsToString( const paths : TStrings ) : String
13204:   Procedure StringToPaths( const pathsString : String; paths : TStrings)
13205:   //Function MacroExpandPath( const aPath : String ) : String
13206: end;
13207:
13208: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13209: begin
13210:   SIRegister_TALMultiPartBaseContent(CL);
13211:   SIRegister_TALMultiPartBaseContents(CL);
13212:   SIRegister_TALMultiPartBaseStream(CL);
13213:   SIRegister_TALMultiPartBaseEncoder(CL);
13214:   SIRegister_TALMultiPartBaseDecoder(CL);
13215:   Function ALMultiPartExtractBoundaryFromContentType( aContentType : AnsiString ) : AnsiString
13216:   Function ALMultiPartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13217:   Function ALMultiPartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13218: end;
13219:
13220: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13221: begin
13222:   TdriveSize', 'record FreeS : Int64; Totals : Int64; end
13223:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13224:   +eger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13225:   Function aAllocPadedMem( Size : Cardinal ) : TObject
13226:   Procedure aFreePadedMem( var P : TObject);
13227:   Procedure aFreePadedMem1( var P : PChar);
13228:   Function aCheckPadedMem( P : Pointer ) : Byte
13229:   Function aGetPadMemSize( P : Pointer ) : Cardinal
13230:   Function aAllocMem( Size : Cardinal ) : Pointer
13231:   Function aStrLen( const Str : PChar ) : Cardinal
13232:   Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal ) : PChar
13233:   Function aStrECopy( Dest : PChar; const Source : PChar ) : PChar
13234:   Function aStrCopy( Dest : PChar; const Source : PChar ) : PChar
13235:   Function aStrEnd( const Str : PChar ) : PChar
13236:   Function aStrScan( const Str : PChar; aChr : Char ) : PChar
13237:   Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal ) : PChar
13238:   Function aPCharLength( const Str : PChar ) : Cardinal
13239:   Function aPCharUpper( Str : PChar ) : PChar
13240:   Function aPCharLower( Str : PChar ) : PChar
13241:   Function aStrCat( Dest : PChar; const Source : PChar ) : PChar
13242:   Function aLastDelimiter( const Delimiters, S : String ) : Integer
13243:   Function aCopyTail( const S : String; Len : Integer ) : String
13244:   Function aInt2Thos( I : Int64 ) : String
13245:   Function aUpperCase( const S : String ) : String
13246:   Function aLowerCase( const S : string ) : String
13247:   Function aCompareText( const S1, S2 : string ) : Integer
13248:   Function aSameText( const S1, S2 : string ) : Boolean
13249:   Function aInt2Str( Value : Int64 ) : String
13250:   Function aStr2Int( const Value : String ) : Int64
13251:   Function aStr2IntDef( const S : string; Default : Int64 ) : Int64
13252:   Function aGetFileExt( const FileName : String ) : String
13253:   Function aGetFilePath( const FileName : String ) : String
13254:   Function aGetFileName( const FileName : String ) : String
13255:   Function aChangeExt( const FileName, Extension : String ) : String
13256:   Function aAdjustLineBreaks( const S : string ) : string
13257:   Function aGetWindowStr( WinHandle : HWND ) : String
13258:   Function aDiskSpace( Drive : String ) : TdriveSize
13259:   Function aFileExists( FileName : String ) : Boolean
13260:   Function aFileSize( FileName : String ) : Int64
13261:   Function aDirectoryExists( const Name : string ) : Boolean
13262:   Function aSysErrorMessage( ErrorCode : Integer ) : string
13263:   Function aShortPathName( const LongName : string ) : string
13264:   Function aGetWindowVer : TWinVerRec
13265:   procedure InitDriveSpacePtr;
13266: end;
13267:
13268: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13269: begin
13270:   aZero', 'LongInt'( 0);
13271:   'makeappDEF', 'LongInt'( - 1);
13272:   'CS_VREDRAW', 'LongInt'( DWORD ( 1 ));
13273:   'CS_HREDRAW', 'LongInt'( DWORD ( 2 ));
13274:   'CS_KEYCVTWINDOW', 'LongInt'( 4);

```

```

13275: 'CS_DBLCLKS','LongInt')( 8);
13276: 'CS_OWNDC','LongWord')($20);
13277: 'CS_CLASSDC','LongWord')($40);
13278: 'CS_PARENTDC','LongWord')($80);
13279: 'CS_NOKEYCVT','LongWord')($100);
13280: 'CS_NOCLOSE','LongWord')($200);
13281: 'CS_SAVEBITS','LongWord')($800);
13282: 'CS_BYTEALIGNCLIENT','LongWord')($1000);
13283: 'CS_BYTEALIGNWINDOW','LongWord')($2000);
13284: 'CS_GLOBALCLASS','LongWord')($4000);
13285: 'CS_IME','LongWord')($10000);
13286: 'CS_DROPSHADOW','LongWord')($20000);
13287: //PPanelFunc, 'TPanelFunc // will not work
13288: TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13289: TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13290: TFontLooks', 'set of TFontLook
13291: TMessagefunc', 'function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13292: Function SetWinClass(const ClassName:String; pMessFunc: TMessagefunc; wcStyle : Integer): Word
13293: Function SetWinClassO( const ClassName : String; pMessFunc : TObjct; wcStyle : Integer): Word
13294: Function SetWinClass( const ClassName : String; pMessFunc : TObjct; wcStyle : Integer): Word
13295: Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13296: Procedure RunMsgLoop( Show : Boolean)
13297: Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13298: Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
ID_Number:Cardinal;hFont:Int):Int;
13299: Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13300: Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13301: Function MakePanel(Left,Top,Width,Height,
hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13302: Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13303: Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13304: Procedure DoInitMakeApp //set first to init formclasscontrol!
13305: end;
13306:
13307: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13308: begin
13309:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13310:   + 'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13311:   TScreenSaverOptions', 'set of TScreenSaverOption
13312:   'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13313:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObjct; previewHwnd: HWND)
13314:   SIRegister_TScreenSaver(CL);
13315:   //Procedure Register
13316:   Procedure SetScreenSaverPassword
13317: end;
13318:
13319: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13320: begin
13321:   FindClass('TOBJECT'),'TXCollection
13322:   SIRegister_EFilerException(CL);
13323:   SIRegister_TXCollectionItem(CL);
13324:   //TXCollectionItemClass', 'class of TXCollectionItem
13325:   SIRegister_TXCollection(CL);
13326:   Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13327:   Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13328:   Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13329:   Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13330:   Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13331:   Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13332: end;
13333:
13334: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13335: begin
13336:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13337:   Procedure xglMapTexCoordToNull
13338:   Procedure xglMapTexCoordToMain
13339:   Procedure xglMapTexCoordToSecond
13340:   Procedure xglMapTexCoordToDual
13341:   Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13342:   Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13343:   Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13344:   Procedure xglBeginUpdate
13345:   Procedure xglEndUpdate
13346:   Procedure xglPushState
13347:   Procedure xglPopState
13348:   Procedure xglForbidSecondTextureUnit
13349:   Procedure xglAllowSecondTextureUnit
13350:   Function xglGetBitWiseMapping : Cardinal
13351: end;
13352:
13353: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13354: begin
13355:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13356:   TBaseListOptions', 'set of TBaseListOption
13357:   SIRegister_TBaseList(CL);
13358:   SIRegister_TBaseVectorList(CL);
13359:   SIRegister_TAffineVectorList(CL);
13360:   SIRegister_TVectorList(CL);

```

```

13361:   SIRegister_TTexPointList(CL);
13362:   SIRegister_TXIntegerList(CL);
13363:   //PSingleArrayList', 'TSingleArrayList // will not work
13364:   SIRegister_TSingleList(CL);
13365:   SIRegister_TByteList(CL);
13366:   SIRegister_TQuaternionList(CL);
13367:   Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13368:   Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13369:   Procedure FastQuickSortLists(startIndex,
endIndex: Integer;refList:TSingleList;objList:TPersistentObjectList);
13370: end;
13371:
13372: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13373: begin
13374:   Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13375:   Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13376:   Procedure ConvertStripToList2(const strip:TAffineVectorList;const
indices:TIntegerList;list:TAffineVectorList);
13377:   Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
indices:TIntegerList;list:TAffineVectorList);
13378:   Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13379:   Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13380:   Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13381:   Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13382:   Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13383:   Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13384:   Procedure RemapIndices( indices, indicesMap : TIntegerList)
13385:   Procedure UnifyTrianglesWinding( indices : TIntegerList)
13386:   Procedure InvertTrianglesWinding( indices : TIntegerList)
13387:   Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13388:   Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
edgesTriangles : TIntegerList) : TIntegerList
13389:   Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13390:   Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
TPersistentObjectList;
13391:   Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13392:   Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13393: end;
13394:
13395: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13396: begin
13397:   Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13398:   Procedure FreeMemAndNil( var P : TObject)
13399:   Function PCharOrNil( const S : string) : PChar
13400:   SIRegister_TJclReferenceMemoryStream(CL);
13401:   FindClass('TObject','EJclVMError
13402: {Function GetVirtualMethodCount( AClass : TClass) : Integer
13403: Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13404: Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13405:   PDynamicIndexList', 'TDynamicIndexList // will not work
13406:   PDynamicAddressList', 'TDynamicAddressList // will not work
13407: Function GetDynamicMethodCount( AClass : TClass) : Integer
13408: Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13409: Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13410: Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13411: Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13412: Function GetInitTable( AClass : TClass) : PTypeInfo
13413:   PFieldEntry', 'TFieldEntry // will not work}
13414:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : Short'
13415:   +'String; end
13416: Function JIsClass( Address : Pointer) : Boolean
13417: Function JIsObject( Address : Pointer) : Boolean
13418: Function GetImplementorOfInterface( const I : IInterface) : TObject
13419:   TDigitCount', 'Integer
13420:   SIRegister_TJclNumericFormat(CL);
13421:   Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13422:   TTextHandler', 'Procedure ( const Text : string)
13423: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13424:   Function JExecute(const
CommandLine:string;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Card;
13425:   Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13426:   Function ReadKey : Char //to and from the DOS console !
13427:   TModuleHandle', 'HINST
13428: //TModuleHandle', 'Pointer
13429: 'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13430:   Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13431:   Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13432:   Procedure UnloadModule( var Module : TModuleHandle)
13433:   Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13434:   Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13435:   Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13436:   Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13437:   FindClass('TObject','EJclConversionError
13438:   Function JStrToBoolean( const S : string) : Boolean
13439:   Function JBooleanToStr( B : Boolean) : string
13440:   Function JIntToBool( I : Integer) : Boolean
13441:   Function JBoolToInt( B : Boolean) : Integer

```



```

13442: 'ListSeparator','String';
13443: 'ListSeparator1','String';
13444: Procedure ListAddItems( var List : string; const Separator, Items : string)
13445: Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13446: Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13447: Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13448: Function ListItemCount( const List, Separator : string) : Integer
13449: Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13450: Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13451: Function ListItemIndex( const List, Separator, Item : string) : Integer
13452: Function SystemToObjectInstance : LongWord
13453: Function IsCompiledWithPackages : Boolean
13454: Function JJclGUIDToString( const GUID : TGUID) : string
13455: Function JJclStringToGUID( const S : string) : TGUID
13456: SIRegister_TJclIntfCriticalSection(CL);
13457: SIRegister_TJclSimpleLog(CL);
13458: Procedure InitSimpleLog( const ALogFileName : string)
13459: end;
13460:
13461: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13462: begin
13463:   FindClass('TObject'),'EJclBorRAException
13464:   TJclBorRADToolKind','( brDelphi, brCppBuilder, brBorlandDevStudio )
13465:   TJclBorRADToolEdition','( deOPEN, dePRO, deSVR )
13466:   TJclBorRADToolEdition','( deSTD, dePRO, deCSS, deARC )
13467:   TJclBorRADToolPath','string
13468:   'SupportedDelphiVersions','LongInt'( 5 or 6 or 7 or 8 or 9 or 10 or 11);
13469:   'SupportedBCBVersions','LongInt'( 5 or 6 or 10 or 11);
13470:   'SupportedBDSVersions','LongInt'( 1 or 2 or 3 or 4 or 5);
13471: BorRADToolRepositoryPagesSection','String 'Repository Pages
13472: BorRADToolRepositoryDialogsPage','String 'Dialogs
13473: BorRADToolRepositoryFormsPage','String 'Forms
13474: BorRADToolRepositoryProjectsPage','String 'Projects
13475: BorRADToolRepositoryDataModulesPage','String 'Data Modules
13476: BorRADToolRepositoryObjectType','String 'Type
13477: BorRADToolRepositoryFormTemplate','String 'FormTemplate
13478: BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13479: BorRADToolRepositoryObjectName','String 'Name
13480: BorRADToolRepositoryObjectPage','String 'Page
13481: BorRADToolRepositoryObjectIcon','String 'Icon
13482: BorRADToolRepositoryObjectDescr','String 'Description
13483: BorRADToolRepositoryObjectAuthor','String 'Author
13484: BorRADToolRepositoryObjectAncestor','String 'Ancestor
13485: BorRADToolRepositoryObjectDesigner','String 'Designer
13486: BorRADToolRepositoryDesignerDfm','String 'dfm
13487: BorRADToolRepositoryDesignerXfm','String 'xfm
13488: BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13489: BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13490: SourceExtensionDelphiPackage','String '.dpk
13491: SourceExtensionBCBPackage','String '.bpb
13492: SourceExtensionDelphiProject','String '.dpr
13493: SourceExtensionBCBProject','String '.bpr
13494: SourceExtensionBDSProject','String '.bdsproj
13495: SourceExtensionDProject','String '.dproj
13496: BinaryExtensionPackage','String '.bpl
13497: BinaryExtensionLibrary','String '.dll
13498: BinaryExtensionExecutable','String '.exe
13499: CompilerExtensionDCP','String '.dcp
13500: CompilerExtensionBPI','String '.bpi
13501: CompilerExtensionLIB','String '.lib
13502: CompilerExtensionTDS','String '.tds
13503: CompilerExtensionMAP','String '.map
13504: CompilerExtensionDRC','String '.drc
13505: CompilerExtensionDEF','String '.def
13506: SourceExtensionCPP','String '.cpp
13507: SourceExtensionH','String '.h
13508: SourceExtensionPAS','String '.pas
13509: SourceExtensionDFM','String '.dfm
13510: SourceExtensionXFM','String '.xfm
13511: SourceDescriptionPAS','String 'Pascal source file
13512: SourceDescriptionCPP','String 'C++ source file
13513: DesignerVCL','String 'VCL
13514: DesignerCLX','String 'CLX
13515: ProjectTypePackage','String 'package
13516: ProjectTypeLibrary','String 'library
13517: ProjectTypeProgram','String 'program
13518: Personality32Bit','String '32 bit
13519: Personality64Bit','String '64 bit
13520: PersonalityDelphi','String 'Delphi
13521: PersonalityDelphiDotNet','String 'Delphi.net
13522: PersonalityBCB','String 'C++Builder
13523: PersonalityCSB','String 'C#Builder
13524: PersonalityVB','String 'Visual Basic
13525: PersonalityDesign','String 'Design
13526: PersonalityUnknown','String 'Unknown personality
13527: PersonalityBDS','String 'Borland Developer Studio
13528: DOFDirectoriesSection','String 'Directories
13529: DOFUnitOutputDirKey','String 'UnitOutputDir
13530: DOFSearchPathName','String 'SearchPath

```

```

13531: DOFConditionals','String 'Conditionals
13532: DOFLinkerSection','String 'Linker
13533: DOFPackagesKey','String 'Packages
13534: DOFCompilerSection','String 'Compiler
13535: DOFPackageNoLinkKey','String 'PackageNoLink
13536: DOFAdditionalSection','String 'Additional
13537: DOFOptionsKey','String 'Options
13538: TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13539:   + 'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13540:   + 'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13541: TJclBorPersonalities', 'set of TJclBorPersonality
13542: TJclBorDesigner', '( bdVCL, bdCLX )
13543: TJclBorDesigners', 'set of TJclBorDesigner
13544: TJclBorPlatform', '( bp32bit, bp64bit )
13545: FindClass('TOBJECT'),'TJclBorRADToolInstallation
13546: SIRegister_TJclBorRADToolInstallationObject(CL);
13547: SIRegister_TJclBorlandOpenHelp(CL);
13548: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13549: TJclHelp2Objects', 'set of TJclHelp2Object
13550: SIRegister_TJclHelp2Manager(CL);
13551: SIRegister_TJclBorRADToolIdeTool(CL);
13552: SIRegister_TJclBorRADToolIdePackages(CL);
13553: SIRegister_IJclCommandLineTool(CL);
13554: FindClass('TOBJECT'),'EJclCommandLineToolError
13555: SIRegister_TJclCommandLineTool(CL);
13556: SIRegister_TJclBorlandCommandLineTool(CL);
13557: SIRegister_TJclBCC32(CL);
13558: SIRegister_TJclDCC32(CL);
13559: TJclDCC', 'TJclDCC32
13560: SIRegister_TJclBpr2Mak(CL);
13561: SIRegister_TJclBorlandMake(CL);
13562: SIRegister_TJclBorRADToolPalette(CL);
13563: SIRegister_TJclBorRADToolRepository(CL);
13564: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
13565: TCommandLineTools', 'set of TCommandLineTool
13566: //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13567: SIRegister_TJclBorRADToolInstallation(CL);
13568: SIRegister_TJclBCBInstallation(CL);
13569: SIRegister_TJclDelphiInstallation(CL);
13570: SIRegister_TJclDCCIL(CL);
13571: SIRegister_TJclBDSInstallation(CL);
13572: TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation ) : Boolean
13573: SIRegister_TJclBorRADToolInstallations(CL);
13574: Function BPLFileName( const BPLPath, PackageFileName : string ) : string
13575: Function BinaryFileName( const OutputPath, ProjectFileName : string ) : string
13576: Function IsDelphiPackage( const FileName : string ) : Boolean
13577: Function IsDelphiProject( const FileName : string ) : Boolean
13578: Function IsBCBPackage( const FileName : string ) : Boolean
13579: Function IsBCBProject( const FileName : string ) : Boolean
13580: Procedure GetDPRFileInfo( const DPRFileName:string;out BinaryExtensio:string;const LibSuffix:PString);
13581: Procedure GetBPRFileInfo( const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13582: Procedure GetDPKFileInfo( const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
Descript:PString;
13583: Procedure GetBPKFileInfo( const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
Descript:PString
13584: function SamePath( const Path1, Path2 : string ) : Boolean;
13585: end;
13586:
13587: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13588: begin
13589:   'ERROR_NO_MORE_FILES', 'LongInt'( 18);
13590:   //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13591:   //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64 ) : Integer
13592:   //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13593:   'LPathSeparator','String '/'
13594:   'LDirDelimiter','String '/'
13595:   'LDirSeparator','String ':'
13596:   'JXPathDevicePrefix','String '\\.\
13597:   'JXPathSeparator','String '\'
13598:   'JXDirDelimiter','String '\'
13599:   'JXDirSeparator','String ';'
13600:   'JXPathUncPrefix','String '\\
13601:   'faNormalFile','LongWord')($00000080);
13602:   //faUnixSpecific',' faSymLink);
13603:   JXTCompactPath', '( cpCenter, cpEnd )
13604:   '_WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13605:     + 'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime : '
13606:     + 'TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13607:   TWIN32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13608:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13609:
13610: Function jxPathAddSeparator( const Path : string ) : string
13611: Function jxPathAddExtension( const Path, Extension : string ) : string
13612: Function jxPathAppend( const Path, Append : string ) : string
13613: Function jxPathBuildRoot( const Drive : Byte ) : string
13614: Function jxPathCanonicalize( const Path : string ) : string
13615: Function jxPathCommonPrefix( const Path1, Path2 : string ) : Integer
13616: Function jxPathCompactPath( const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13617: Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)

```

```

13618: Function jxPathExtractFileDirFixed( const S : string ) : string
13619: Function jxPathExtractFileNameNoExt( const Path : string ) : string
13620: Function jxPathExtractPathDepth( const Path : string; Depth : Integer ) : string
13621: Function jxPathGetDepth( const Path : string ) : Integer
13622: Function jxPathGetLongName( const Path : string ) : string
13623: Function jxPathGetShortName( const Path : string ) : string
13624: Function jxPathGetLongName( const Path : string ) : string
13625: Function jxPathGetShortName( const Path : string ) : string
13626: Function jxPathGetRelativePath( Origin, Destination : string ) : string
13627: Function jxPathGetTempPath : string
13628: Function jxPathIsAbsolute( const Path : string ) : Boolean
13629: Function jxPathIsChild( const Path, Base : string ) : Boolean
13630: Function jxPathIsDiskDevice( const Path : string ) : Boolean
13631: Function jxPathIsUNC( const Path : string ) : Boolean
13632: Function jxPathRemoveSeparator( const Path : string ) : string
13633: Function jxPathRemoveExtension( const Path : string ) : string
13634: Function jxPathGetPhysicalPath( const LocalizedPath : string ) : string
13635: Function jxPathGetLocalizedPath( const PhysicalPath : string ) : string
13636: JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13637: JxTFileListOptions', 'set of TFileListOption
13638: JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13639: TFileHandler', 'Procedure ( const FileName : string)
13640: TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13641: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings ) : Boolean
13642: //Function AdvBuildFileList( const Path : string; const Mask : Integer; const Files : TStrings; const
AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
FileMatchFunc:TFileMatchFunc):Bool;
13643: Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int ) : Boolean
13644: Function jxIsFileAttributeMatch( FileAttributes, RejectedAttributes, RequiredAttributes: Int ): Boolean;
13645: Function jxFileAttributesStr( const FileInfo : TSearchRec ) : string
13646: Function jxIsFileNameMatch( FileName:string;const Mask:string;const CaseSensitive:Boolean): Boolean;
13647: Procedure jxEnumFiles( const Path:string; HandleFile:TFileHandlerEx;
RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13648: Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13649: Procedure jxCreateEmptyFile( const FileName : string)
13650: Function jxCloseVolume( var Volume : THandle ) : Boolean
13651: Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean ) : Boolean
13652: Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13653: Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13654: Function jxDelTree( const Path : string ) : Boolean
13655: //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13656: Function jxDiskInDrive( Drive : Char ) : Boolean
13657: Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean ) : Boolean
13658: Function jxFileCreateTemp( var Prefix : string ) : THandle
13659: Function jxFileBackup( const FileName : string; Move : Boolean ) : Boolean
13660: Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13661: Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
13662: Function jxFileExists( const FileName : string ) : Boolean
13663: Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13664: Function jxFileRestore( const FileName : string ) : Boolean
13665: Function jxGetBackupFileName( const FileName : string ) : string
13666: Function jxIsBackupFileName( const FileName : string ) : Boolean
13667: Function jxFileGetDisplayName( const FileName : string ) : string
13668: Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean ) : string
13669: Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean ) : string
13670: Function jxFileGetSize( const FileName : string ) : Int64
13671: Function jxFileGetTempName( const Prefix : string ) : string
13672: Function jxFileGetTypeNames( const FileName : string ) : string
13673: Function jxFindUnusedFileName( FileName:string; const FileExt : string; NumberPrefix : string ) : string
13674: Function jxForceDirectories( Name : string ) : Boolean
13675: Function jxGetDirectorySize( const Path : string ) : Int64
13676: Function jxGetDriveTypeStr( const Drive : Char ) : string
13677: Function jxGetFileAgeCoherence( const FileName : string ) : Boolean
13678: Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13679: Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13680: Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec ) : Boolean;
13681: Function jxGetFileInformation1( const FileName : string ) : TSearchRec;
13682: //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
ResolveSymLinks:Boolean):Integer
13683: Function jxGetFileLastWrite( const FName : string ) : TFileTime;
13684: Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13685: Function jxGetFileLastAccess( const FName : string ) : TFileTime;
13686: Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13687: Function jxGetFileCreation( const FName : string ) : TFileTime;
13688: Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13689: Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13690: Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13691: Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean ) : Integer;
13692: Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13693: Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13694: Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13695: Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13696: Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13697: Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13698: Function jxGetModulePath( const Module : HMODULE ) : string
13699: Function jxGetSizeOfFile( const FileName : string ) : Int64;
13700: Function jxGetSizeOfFile1( const FileInfo : TSearchRec ) : Int64;
13701: Function jxGetSizeOfFile2( Handle : THandle ) : Int64;

```

```

13702: Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13703: Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13704: Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13705: Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13706: Function jxOpenVolume( const Drive : Char) : THandle
13707: Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13708: Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13709: Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13710: Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13711: Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13712: Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13713: Procedure jxShredFile( const FileName : string; Times : Integer)
13714: Function jxUnlockVolume( var Handle : THandle) : Boolean
13715: Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13716: Function jxSymbolicLinkTarget( const Name : string) : string
13717: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13718: SIRegister_TJclCustomFileAttrMask(CL);
13719: SIRegister_TJclFileAttributeMask(CL);
13720: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13721: + 'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13722: TFileSearchOptions', 'set of TFileSearchOption
13723: TFileSearchTaskID', 'Integer
13724: TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearch'
13725: + 'hTaskID; const Aborted : Boolean)
13726: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13727: SIRegister_IJclFileEnumerator(CL);
13728: SIRegister_TJclFileEnumerator(CL);
13729: Function jxFileSearch : IJclFileEnumerator
13730: JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched, ffPreRelease, ffPrivateBuild, ffSpecialBuild )
13731: JxTFileFlags', 'set of TFileFlag
13732: FindClass('TOBJECT'), 'EJclFileVersionInfoError
13733: SIRegister_TJclFileVersionInfo(CL);
13734: Function jxOSIdentToString( const OSIdent : DWORD) : string
13735: Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13736: Function jxVersionResourceAvailable( const FileName : string) : Boolean
13737: TFileVersionFormat', '( vfMajorMinor, vfFull )
13738: Function jxFormatVersionString( const HiV, LoV : Word) : string;
13739: Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13740: //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo; VersionFormat: TFileVersionFormat):str;
13741: //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13742: //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
Revision:Word);
13743: //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13744: Function jxVersionFixedFileInfoToString( const FileName : string; VersionFormat : TFileVersionFormat; const
NotAvailableText : string) : string
13745: SIRegister_TJclTempFileStream(CL);
13746: FindClass('TOBJECT'), 'TJclCustomFileMapping
13747: SIRegister_TJclFileMapView(CL);
13748: TJclFileMappingRoundOffset', '( rvDown, rvUp )
13749: SIRegister_TJclCustomFileMapping(CL);
13750: SIRegister_TJclFileMapping(CL);
13751: SIRegister_TJclSwapFileMapping(CL);
13752: SIRegister_TJclFileMappingStream(CL);
13753: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13754: //PPCharArray', '^TPCharArray // will not work
13755: SIRegister_TJclMappedTextReader(CL);
13756: SIRegister_TJclFileMaskComparator(CL);
13757: FindClass('TOBJECT'), 'EJclPathError
13758: FindClass('TOBJECT'), 'EJclFileUtilsError
13759: FindClass('TOBJECT'), 'EJclTempFileStreamError
13760: FindClass('TOBJECT'), 'EJclTempFileStreamError
13761: FindClass('TOBJECT'), 'EJclFileMappingError
13762: FindClass('TOBJECT'), 'EJclFileMapViewError
13763: Function jxPathGetLongName2( const Path : string) : string
13764: Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13765: Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13766: Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13767: Function jxWin32RestoreFile( const FileName : string) : Boolean
13768: Function jxSamePath( const Path1, Path2 : string) : Boolean
13769: Procedure jxPathListAddItems( var List : string; const Items : string)
13770: Procedure jxPathListIncludeItems( var List : string; const Items : string)
13771: Procedure jxPathListDelItems( var List : string; const Items : string)
13772: Procedure jxPathListDelItem( var List : string; const Index : Integer)
13773: Function jxPathListItemCount( const List : string) : Integer
13774: Function jxPathListItemGetItem( const List : string; const Index : Integer) : string
13775: Procedure jxPathListItemSetItem( var List : string; const Index : Integer; const Value : string)
13776: Function jxPathListItemIndex( const List, Item : string) : Integer
13777: Function jxParamName(Idx: Int; const Separator: string; const AllowedPrefixChars: string; TrimName: Bool): string;
13778: Function jxParamValue(Idx : Integer; const Separator : string; TrimValue : Boolean) : string;
13779: Function jxParamValue1( const SearchName: string; const Separator : string; CaseSensitive : Boolean; const
AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13780: Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
AllowedPrefixCharacters : string) : Integer
13781: end;
13782:
13783: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13784: begin
13785: 'UTF8FileHeader', 'String # $ef$ $bb$ $bf);
13786: Function lCompareFileNames( const Filename1, Filename2 : string) : integer

```



```

13787: Function lCompareFileNamesIgnoreCase( const Filename1, Filename2 : string ) : integer
13788: Function lCompareFileNames( const Filename1, Filename2 : string; ResolveLinks : boolean ) : integer
13789: Function lCompareFileNames( Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13790: Function lFilenameIsAbsolute( const TheFilename : string ) : boolean
13791: Function lFilenameIsWinAbsolute( const TheFilename : string ) : boolean
13792: Function lFilenameIsUnixAbsolute( const TheFilename : string ) : boolean
13793: Procedure lCheckIfFileIsExecutable( const AFilename : string)
13794: Procedure lCheckIfFileIsSymlink( const AFilename : string)
13795: Function lFileIsReadable( const AFilename : string ) : boolean
13796: Function lFileIsWritable( const AFilename : string ) : boolean
13797: Function lFileIsText( const AFilename : string ) : boolean
13798: Function lFileIsText( const AFilename : string; out FileReadable : boolean ) : boolean
13799: Function lFileIsExecutable( const AFilename : string ) : boolean
13800: Function lFileIsSymlink( const AFilename : string ) : boolean
13801: Function lFileIsHardLink( const AFilename : string ) : boolean
13802: Function lFileSize( const Filename : string ) : int64;
13803: Function lGetFileDescription( const AFilename : string ) : string
13804: Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean ) : string
13805: Function lTryReadAllLinks( const Filename : string ) : string
13806: Function lDirPathExists( const FileName : string ) : Boolean
13807: Function lForceDirectory( DirectoryName : string ) : boolean
13808: Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean ) : boolean
13809: Function lProgramDirectory : string
13810: Function lDirectoryIsWritable( const DirectoryName : string ) : boolean
13811: Function lExtractFileNameOnly( const AFilename : string ) : string
13812: Function lExtractFileNameWithoutExt( const AFilename : string ) : string
13813: Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean ) : integer;
13814: Function lCompareFileExt( const Filename, Ext : string ) : integer;
13815: Function lFilenameIsPascalUnit( const Filename : string ) : boolean
13816: Function lAppendPathDelim( const Path : string ) : string
13817: Function lChompPathDelim( const Path : string ) : string
13818: Function lTrimFilename( const AFilename : string ) : string
13819: Function lCleanAndExpandFilename( const Filename : string ) : string
13820: Function lCleanAndExpandDirectory( const Filename : string ) : string
13821: Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string ) : string
13822: Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
AlwaysRequireSharedBaseFolder : Boolean ) : string
13823: Function lCreateAbsolutePath( const Filename, BaseDirectory : string ) : string
13824: Function lFileIsInPath( const Filename, Path : string ) : boolean
13825: Function lFileIsInDirectory( const Filename, Directory : string ) : boolean
13826: TSearchFileInPathFlag, '( sffDontSearchInBasePath, sffSearchLoUpCase )
13827: TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13828: 'AllDirectoryEntriesMask','String '*
13829: Function lGetAllFilesMask : string
13830: Function lGetExeExt : string
13831: Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
TSearchFileInPathFlags ) : string
13832: Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
TSearchFileInPathFlags ) : TStringList
13833: Function lFindDiskFilename( const Filename : string ) : string
13834: Function lFindDiskFileCaseInsensitive( const Filename : string ) : string
13835: Function lFindDefaultExecutablePath( const Executable : string; const BaseDir : string):string
13836: Function lGetDarwinSystemFilename( Filename : string ) : string
13837: SIRegister_TFileIterator(CL);
13838: TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13839: TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13840: TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13841: SIRegister_TFileSearcher(CL);
13842: Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13843: Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean ) : TStringList
13844: // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13845: // TCopyFileFlags', 'set of TCopyFileFlag
13846: Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags ) : boolean
13847: Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean ) : boolean
13848: Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags ) : Boolean
13849: Function lReadFileToString( const Filename : string ) : string
13850: Function lGetTempFilename( const Directory, Prefix : string ) : string
13851: {Function NeedRTLAnsi : boolean
13852: Procedure SetNeedRTLAnsi( NewValue : boolean)
13853: Function UTF8ToSys( const s : string ) : string
13854: Function SysToUTF8( const s : string ) : string
13855: Function ConsoleToUTF8( const s : string ) : string
13856: Function UTF8ToConsole( const s : string ) : string}
13857: Function FileExistsUTF8( const Filename : string ) : boolean
13858: Function FileAgeUTF8( const FileName : string ) : Longint
13859: Function DirectoryExistsUTF8( const Directory : string ) : Boolean
13860: Function ExpandFileNameUTF8( const FileName : string ) : string
13861: Function ExpandUNCFileNameUTF8( const FileName : string ) : string
13862: Function ExtractShortPathNameUTF8( const FileName : String ) : String
13863: Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec ) : Longint
13864: Function FindNextUTF8( var Rslt : TSearchRec ) : Longint
13865: Procedure FindCloseUTF8( var F : TSearchRec)
13866: Function FileSetDateUTF8( const FileName : String; Age : Longint ) : Longint
13867: Function FileGetAttrUTF8( const FileName : String ) : Longint
13868: Function FileSetAttrUTF8( const FileName : String; Attr : longint ) : Longint
13869: Function DeleteFileUTF8( const FileName : String ) : Boolean
13870: Function RenameFileUTF8( const OldName, NewName : String ) : Boolean
13871: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean ) : String
13872: Function FileIsReadOnlyUTF8( const FileName : String ) : Boolean

```

```

13873: Function GetCurrentDirUTF8 : String
13874: Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13875: Function CreateDirUTF8( const NewDir : String) : Boolean
13876: Function RemoveDirUTF8( const Dir : String) : Boolean
13877: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13878: Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13879: Function FileCreateUTF8( const FileName : string) : THandle;
13880: Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13881: Function ParamStrUTF8( Param : Integer) : string
13882: Function GetEnvironmentStringUTF8( Index : Integer) : string
13883: Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13884: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13885: Function GetAppConfigFileUTF8(Global: Boolean; SubDir: boolean; CreateDir : boolean) : string
13886: Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13887: end;
13888:
13889: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13890: begin
13891:   //VK_F23 = 134;
13892:   //{$EXTERNALSYM VK_F24}
13893:   //VK_F24 = 135;
13894:   TVirtualKeyCode, 'Integer
13895:   'VK_MOUSEWHEELUP', 'integer'(134);
13896:   'VK_MOUSEWHEELDOWN', 'integer'(135);
13897: Function gllsKeyDown( c : Char) : Boolean;
13898: Function gllsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13899: Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13900: Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13901: Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13902: Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13903: Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13904: end;
13905:
13906: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13907: begin
13908:   TGLPoint, 'TPoint
13909:   //PGLPoint', '^TGLPoint // will not work
13910:   TGLRect, 'TRect
13911:   //PGLRect', '^TGLRect // will not work
13912:   TDelphiColor, 'TColor
13913:   TGLPicture, 'TPicture
13914:   TGLGraphic, 'TGraphic
13915:   TGLBitmap, 'TBitmap
13916:   //TGraphicClass', 'class of TGraphic
13917:   TGLTextLayout, '( tlTop, tlCenter, tlBottom )
13918:   TGLMouseButton, '( mbLeft, mbRight, mbMiddle )
13919:   TGLMouseEvent, 'Procedure ( Sender : TObject; Button : TGLMouse'
13920:   + 'Button; Shift : TShiftState; X, Y : Integer)
13921:   TGLMouseMoveEvent, 'TMouseMoveEvent
13922:   TGLKeyEvent, 'TKeyEvent
13923:   TGLKeyPressEvent, 'TKeyPressEvent
13924:   EGLOSError, 'EWin32Error
13925:   EGLOSError, 'EWin32Error
13926:   EGLOSError, 'EOSError
13927:   'glsAllFilter', 'string'All // sAllFilter
13928: Function GLPoint( const x, y : Integer) : TGLPoint
13929: Function GLRGB( const r, g, b : Byte) : TColor
13930: Function GLColorToRGB( color : TColor) : TColor
13931: Function GLGetRValue( rgb : DWORD) : Byte
13932: Function GLGetGValue( rgb : DWORD) : Byte
13933: Function GLGetBValue( rgb : DWORD) : Byte
13934: Procedure GLInitWinColors
13935: Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
13936: Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
13937: Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
13938: Procedure GLInformationDlg( const msg : String)
13939: Function GLQuestionDlg( const msg : String) : Boolean
13940: Function GLInputDialog( const aCaption, aPrompt, aDefault : String) : String
13941: Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
13942: Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
13943: Function GLApplicationTerminated : Boolean
13944: Procedure GLRaiseLastError
13945: Procedure GLFreeAndNil( var anObject: TObject)
13946: Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
13947: Function GLGetCurrentColorDepth : Integer
13948: Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
13949: Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
13950: Procedure GLSleep( length : Cardinal)
13951: Procedure GLQueryPerformanceCounter( var val : Int64)
13952: Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
13953: Function GLStartPrecisionTimer : Int64
13954: Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
13955: Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
13956: Function GLRDTSC : Int64
13957: Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
13958: Function GLOKMessageBox( const Text, Caption : string) : Integer
13959: Procedure GLShowHTMLUrl( Url : String)
13960: Procedure GLShowCursor( AShow : boolean)
13961: Procedure GLSetCursorPos( AScreenX, AScreenY : integer)

```

```

13962: Procedure GLGetCursorPos( var point : TGLPoint)
13963: Function GLGetScreenWidth : integer
13964: Function GLGetScreenHeight : integer
13965: Function GLGetTickCount : int64
13966: function RemoveSpaces(const str : String) : String;
13967: TNormalMapSpace', '( nmsObject, nmsTangent )
13968: Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
13969: Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
13970: Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
TAffineVectorList; Colors : TVectorList)
13971: Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
HiTexCoords:TAffineVectorList):TGLBitmap
13972: Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
LoTexCoords, Tangents, BiNormals : TAffineVectorList ) : TGLBitmap
13973: end;
13974:
13975: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
13976: begin
13977:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
13978:   // PGLStarRecord', '^TGLStarRecord // will not work
13979:   Function StarRecordPositionZUp( const starRecord : TGLStarRecord ) : TAffineVector
13980:   Function StarRecordPositionYUp( const starRecord : TGLStarRecord ) : TAffineVector
13981:   Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single ) : TVector
13982: end;
13983:
13984:
13985: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
13986: begin
13987:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
13988:   //PAABB', '^TAABB // will not work
13989:   TBSphere', 'record Center : TAffineVector; Radius : single; end
13990:   TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
13991:   TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
13992:   Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox ) : THmgBoundingBox
13993:   Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
13994:   Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
13995:   Procedure SetAABB( var bb : TAABB; const v : TVector)
13996:   Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
13997:   Procedure AABBBTransform( var bb : TAABB; const m : TMatrix)
13998:   Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
13999:   Function BBMinX( const c : THmgBoundingBox ) : Single
14000:   Function BBMaxX( const c : THmgBoundingBox ) : Single
14001:   Function BBMinY( const c : THmgBoundingBox ) : Single
14002:   Function BBMaxY( const c : THmgBoundingBox ) : Single
14003:   Function BBMinZ( const c : THmgBoundingBox ) : Single
14004:   Function BBMaxZ( const c : THmgBoundingBox ) : Single
14005:   Procedure AABBIinclude( var bb : TAABB; const p : TAffineVector)
14006:   Procedure AABBFfromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14007:   Function AABBIIntersection( const aabb1, aabb2 : TAABB ) : TAABB
14008:   Function BBToAABB( const aabb : THmgBoundingBox ) : TAABB
14009:   Function AABBTtoBB( const anAABB : TAABB ) : THmgBoundingBox;
14010:   Function AABBTtoBB1( const anAABB : TAABB; const m : TMatrix ) : THmgBoundingBox;
14011:   Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14012:   Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14013:   Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix ) : Boolean;
14014:   Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB ) : Boolean
14015:   Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB ) : Boolean
14016:   Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14017:   Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14018:   Function PointInAABB( const p : TAffineVector; const aabb : TAABB ) : Boolean;
14019:   Function PointInAABB1( const p : TVector; const aabb : TAABB ) : Boolean;
14020:   Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB ) : boolean
14021:   Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector ) : boolean
14022:   Procedure ExtractAABBCorners( const AABBB : TAABB; var AABBCorners : TAABBCorners)
14023:   Procedure AABBTtoBSphere( const AABBB : TAABB; var BSphere : TBSphere)
14024:   Procedure BSphereToAABB( const BSphere : TBSphere; var AABBB : TAABB);
14025:   Function BSphereToAABB1( const center : TAffineVector; radius : Single ) : TAABB;
14026:   Function BSphereToAABB2( const center : TVector; radius : Single ) : TAABB;
14027:   Function AABBBContainsAABB( const mainAABB, testAABB : TAABB ) : TSpaceContains
14028:   Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB ) : TSpaceContains
14029:   Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere ) : TSpaceContains
14030:   Function AABBBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere ) : TSpaceContains
14031:   Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
testBSphere:TBSphere):TSpaceContains
14032:   Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere ) : TSpaceContains
14033:   Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB ) : TSpaceContains
14034:   Function ClipToAABB( const v : TAffineVector; const AABBB : TAABB ) : TAffineVector
14035:   Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere ) : boolean
14036:   Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14037:   Function AABBTtoClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
viewportSizeY:Int):TClipRect
14038: end;
14039:
14040: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14041: begin
14042:   Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14043:   Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14044:   Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14045:   Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);

```

```

14046: Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14047: Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14048: Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14049: Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14050: Procedure Spherical_Cartesian2( const r, theta, phi : single; var x, y, z : single; var ierr : integer);
14051: Procedure Spherical_Cartesian3( const r, theta, phi : double; var x, y, z : double; var ierr : integer);
14052: Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14053: Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14054: Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14055: Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14056: Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14057: Procedure ProlateSpheroidal_Cartesian2( const xi, eta, phi, a : single; var x, y, z : single; var ierr : integer);
14058: Procedure ProlateSpheroidal_Cartesian3( const xi, eta, phi, a : double; var x, y, z : double; var ierr : integer);
14059: Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14060: Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14061: Procedure OblateSpheroidal_Cartesian2( const xi, eta, phi, a : single; var x, y, z : single; var ierr : integer);
14062: Procedure OblateSpheroidal_Cartesian3( const xi, eta, phi, a : double; var x, y, z : double; var ierr : integer);
14063: Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14064: Procedure BipolarCylindrical_Cartesian1( const u, v, z1, a : double; var x, y, z : double);
14065: Procedure BipolarCylindrical_Cartesian2( const u, v, z1, a : single; var x, y, z : single; var ierr : integer);
14066: Procedure BipolarCylindrical_Cartesian3( const u, v, z1, a : double; var x, y, z : double; var ierr : integer);
14067: end;
14068:
14069: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14070: begin
14071:   'EPSILON', 'Single').setExtended( 1e-40);
14072:   'EPSILON2', 'Single').setExtended( 1e-30); }
14073: TRenderContextClippingInfo, 'record origin : TVector; clippingD'
14074:   + 'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14075: THmgPlane', 'TVector
14076:   TDoubleHmgPlane', 'THomogeneousDblVector
14077: {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14078:   + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14079:   + ', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW } }
14080: TSingleArray', 'array of Single
14081: TTransformations', 'array [0..15] of Single)
14082: TPackedRotationMatrix', 'array [0..2] of Smallint)
14083: TVertex', 'TAffineVector
14084: //TVectorGL', 'THomogeneousFltVector
14085: //TMatrixGL', 'THomogeneousFltMatrix
14086: // TPackedRotationMatrix = array [0..2] of SmallInt;
14087: Function glTexPointMake( const s, t : Single) : TTexPoint
14088: Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14089: Function glAffineVectorMakel( const v : TVectorGL) : TAffineVector;
14090: Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14091: Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14092: Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14093: Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14094: Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14095: Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14096: Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14097: Function glVectorMakel( const x, y, z : Single; w : Single) : TVectorGL;
14098: Function glPointMake( const x, y, z : Single) : TVectorGL;
14099: Function glPointMakel( const v : TAffineVector) : TVectorGL;
14100: Function glPointMake2( const v : TVectorGL) : TVectorGL;
14101: Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14102: Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14103: Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14104: Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14105: Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14106: Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14107: Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14108: Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14109: Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14110: Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14111: Procedure glRstVector( var v : TAffineVector);
14112: Procedure glRstVector1( var v : TVectorGL);
14113: Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14114: Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14115: //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14116: Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14117: Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14118: Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14119: Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14120: Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14121: Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14122: Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14123: Procedure glAddVector10( var v : TAffineVector; const f : Single);
14124: Procedure glAddVector11( var v : TVectorGL; const f : Single);
14125: //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Integer;dest:PTexPointArray);
14126: //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14127: //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
PAffineVectorArray);
14128: Function glVectorSubtract( const v1, v2 : TAffineVector) : TAffineVector;
14129: Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14130: Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14131: Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);

```



```

14132: Function glVectorSubtract4( const V1, V2 : TVectorGL ) : TVectorGL;
14133: Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14134: Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14135: Function glVectorSubtract7( const v1 : TAffineVector; delta : Single ) : TAffineVector;
14136: Function glVectorSubtract8( const v1 : TVectorGL; delta : Single ) : TVectorGL;
14137: Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14138: Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14139: Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14140: //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14141: Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single ) : TTexPoint
14142: Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single ) : TAffineVector;
14143: Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single ) : TAffineVector;
14144: Procedure glVectorCombine34( const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14145: Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14146: Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14147: Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single ) : TVectorGL;
14148: Function glVectorCombine8( const V1 : TVectorGL; const V2 : TAffineVector; const F1,F2:Single): TVectorGL;
14149: Procedure glVectorCombine9( const V1:TVectorGL; const V2:TAffineVector; const F1,F2:Single; var vr:TVectorGL);
14150: Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14151: Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14152: Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single ) : TVectorGL;
14153: Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14154: Function glVectorDotProduct( const V1, V2 : TAffineVector ) : Single;
14155: Function glVectorDotProduct1( const V1, V2 : TVectorGL ) : Single;
14156: Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector ) : Single;
14157: Function glPointProject( const p, origin, direction : TAffineVector ) : Single;
14158: Function glPointProject1( const p, origin, direction : TVectorGL ) : Single;
14159: Function glVectorCrossProduct( const V1, V2 : TAffineVector ) : TAffineVector;
14160: Function glVectorCrossProduct1( const V1, V2 : TVectorGL ) : TVectorGL;
14161: Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14162: Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14163: Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14164: Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14165: Function glLerp( const start, stop, t : Single ) : Single
14166: Function glAngleLerp( start, stop, t : Single ) : Single
14167: Function glDistanceBetweenAngles( angle1, angle2 : Single ) : Single
14168: Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single ) : TTexPoint;
14169: Function glVectorLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14170: Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14171: Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single ) : TVectorGL;
14172: Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14173: Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14174: Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single ) : TAffineVector;
14175: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14176: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
PAffineVectorArray);
14177: Function glVectorLength( const x, y : Single ) : Single;
14178: Function glVectorLength1( const x, y, z : Single ) : Single;
14179: Function glVectorLength2( const v : TAffineVector ) : Single;
14180: Function glVectorLength3( const v : TVectorGL ) : Single;
14181: Function glVectorLength4( const v : array of Single ) : Single;
14182: Function glVectorNorm( const x, y : Single ) : Single;
14183: Function glVectorNorm1( const v : TAffineVector ) : Single;
14184: Function glVectorNorm2( const v : TVectorGL ) : Single;
14185: Function glVectorNorm3( var V : array of Single ) : Single;
14186: Procedure glNormalizeVector( var v : TAffineVector);
14187: Procedure glNormalizeVector1( var v : TVectorGL);
14188: Function glVectorNormalize( const v : TAffineVector ) : TAffineVector;
14189: Function glVectorNormalize1( const v : TVectorGL ) : TVectorGL;
14190: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14191: Function glVectorAngleCosine( const V1, V2 : TAffineVector ) : Single
14192: Function glVectorNegate( const v : TAffineVector ) : TAffineVector;
14193: Function glVectorNegate1( const v : TVectorGL ) : TVectorGL;
14194: Procedure glNegateVector( var V : TAffineVector);
14195: Procedure glNegateVector2( var V : TVectorGL);
14196: Procedure glNegateVector3( var V : array of Single);
14197: Procedure glScaleVector( var v : TAffineVector; factor : Single);
14198: Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14199: Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14200: Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14201: Function glVectorScale( const v : TAffineVector; factor : Single ) : TAffineVector;
14202: Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14203: Function glVectorScale2( const v : TVectorGL; factor : Single ) : TVectorGL;
14204: Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14205: Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14206: Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14207: Function glVectorEquals( const V1, V2 : TVectorGL ) : Boolean;
14208: Function glVectorEquals1( const V1, V2 : TAffineVector ) : Boolean;
14209: Function glAffineVectorEquals( const V1, V2 : TVectorGL ) : Boolean;
14210: Function glVectorIsNull( const v : TVectorGL ) : Boolean;
14211: Function glVectorIsNull1( const v : TAffineVector ) : Boolean;
14212: Function glVectorSpacing( const v1, v2 : TTexPoint ) : Single;
14213: Function glVectorSpacing1( const v1, v2 : TAffineVector ) : Single;
14214: Function glVectorSpacing2( const v1, v2 : TVectorGL ) : Single;
14215: Function glVectorDistance( const v1, v2 : TAffineVector ) : Single;
14216: Function glVectorDistance1( const v1, v2 : TVectorGL ) : Single;
14217: Function glVectorDistance2( const v1, v2 : TAffineVector ) : Single;
14218: Function glVectorDistance21( const v1, v2 : TVectorGL ) : Single;
14219: Function glVectorPerpendicular( const V, N : TAffineVector ) : TAffineVector

```

```

14220: Function glVectorReflect( const V, N : TAffineVector ) : TAffineVector
14221: Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14222: Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14223: Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14224: Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single ) : TAffineVector;
14225: Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single ) : TAffineVector;
14226: Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14227: Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single ) : TAffineVector;
14228: Procedure glAbsVector( var v : TVectorGL);
14229: Procedure glAbsVector1( var v : TAffineVector);
14230: Function glVectorAbs( const v : TVectorGL ) : TVectorGL;
14231: Function glVectorAbs1( const v : TAffineVector ) : TAffineVector;
14232: Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14233: Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14234: Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14235: Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14236: Function glCreateScaleMatrix( const v : TAffineVector ) : TMatrixGL;
14237: Function glCreateScaleMatrix1( const v : TVectorGL ) : TMatrixGL;
14238: Function glCreateTranslationMatrix( const V : TAffineVector ) : TMatrixGL;
14239: Function glCreateTranslationMatrix1( const V : TVectorGL ) : TMatrixGL;
14240: Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL ) : TMatrixGL;
14241: Function glCreateRotationMatrixX( const sine, cosine : Single ) : TMatrixGL;
14242: Function glCreateRotationMatrixX1( const angle : Single ) : TMatrixGL;
14243: Function glCreateRotationMatrixY( const sine, cosine : Single ) : TMatrixGL;
14244: Function glCreateRotationMatrixY1( const angle : Single ) : TMatrixGL;
14245: Function glCreateRotationMatrixZ( const sine, cosine : Single ) : TMatrixGL;
14246: Function glCreateRotationMatrixZ1( const angle : Single ) : TMatrixGL;
14247: Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single ) : TMatrixGL;
14248: Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single ) : TMatrixGL;
14249: Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14250: Function glMatrixMultiply( const M1, M2 : TAffineMatrix ) : TAffineMatrix;
14251: Function glMatrixMultiply1( const M1, M2 : TMatrixGL ) : TMatrixGL;
14252: Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14253: Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL ) : TVectorGL;
14254: Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix ) : TVectorGL;
14255: Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL ) : TAffineVector;
14256: Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix ) : TAffineVector;
14257: Function glMatrixDeterminant( const M : TAffineMatrix ) : Single;
14258: Function glMatrixDeterminant1( const M : TMatrixGL ) : Single;
14259: Procedure glAdjointMatrix( var M : TMatrixGL);
14260: Procedure glAdjointMatrix1( var M : TAffineMatrix);
14261: Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14262: Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14263: Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14264: Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14265: Procedure glNormalizeMatrix( var M : TMatrixGL);
14266: Procedure glTransposeMatrix( var M : TAffineMatrix);
14267: Procedure glTransposeMatrix1( var M : TMatrixGL);
14268: Procedure glInvertMatrix( var M : TMatrixGL);
14269: Procedure glInvertMatrix1( var M : TAffineMatrix);
14270: Function glAnglePreservingMatrixInvert( const mat : TMatrixGL ) : TMatrixGL
14271: Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations ) : Boolean
14272: Function glPlaneMake( const p1, p2, p3 : TAffineVector ) : THmgPlane;
14273: Function glPlaneMake1( const p1, p2, p3 : TVectorGL ) : THmgPlane;
14274: Function glPlaneMake2( const point, normal : TAffineVector ) : THmgPlane;
14275: Function glPlaneMake3( const point, normal : TVectorGL ) : THmgPlane;
14276: Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14277: Procedure glNormalizePlane( var plane : THmgPlane)
14278: Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector ) : Single;
14279: Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL ) : Single;
14280: Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector ) : TAffineVector;
14281: Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14282: Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14283: Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL ) : Boolean;
14284: Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector ) : Boolean;
14285: Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL ) : Single;
14286: Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector ) : Single;
14287: Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14288: Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector ) : single
14289: Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector ) : TAffineVector
14290: Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector ) : Single
14291: Procedure SgsegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
Segment0Closest, Segment1Closest : TAffineVector)
14292: Function glSegmentSegmentDistance( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector ) : single
14293: TEulerOrder', '( eulXYZ, eulXZY, eulYZX, eulZYX, eulZXY, eulZYX)
14294: Function glQuaternionMake( const Imag : array of Single; Real : Single ) : TQuaternion
14295: Function glQuaternionConjugate( const Q : TQuaternion ) : TQuaternion
14296: Function glQuaternionMagnitude( const Q : TQuaternion ) : Single
14297: Procedure glNormalizeQuaternion( var Q : TQuaternion)
14298: Function glQuaternionFromPoints( const V1, V2 : TAffineVector ) : TQuaternion
14299: Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14300: Function glQuaternionFromMatrix( const mat : TMatrixGL ) : TQuaternion
14301: Function glQuaternionToMatrix( quat : TQuaternion ) : TMatrixGL
14302: Function glQuaternionToAffineMatrix( quat : TQuaternion ) : TAffineMatrix
14303: Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector ) : TQuaternion
14304: Function glQuaternionFromRollPitchYaw( const r, p, y : Single ) : TQuaternion
14305: Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder ) : TQuaternion
14306: Function glQuaternionMultiply( const qL, qR : TQuaternion ) : TQuaternion
14307: Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single ) : TQuaternion;

```

```

14308: Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14309: Function glLnXP1( X : Extended) : Extended
14310: Function glLog10( X : Extended) : Extended
14311: Function glLog2( X : Extended) : Extended;
14312: Function glLog21( X : Single) : Single;
14313: Function glLogN( Base, X : Extended) : Extended
14314: Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14315: Function glPower( const Base, Exponent : Single) : Single;
14316: Function glPower1( Base : Single; Exponent : Integer) : Single;
14317: Function glDegToRad( const Degrees : Extended) : Extended;
14318: Function glDegToRad1( const Degrees : Single) : Single;
14319: Function glRadToDeg( const Radians : Extended) : Extended;
14320: Function glRadToDeg1( const Radians : Single) : Single;
14321: Function glNormalizeAngle( angle : Single) : Single
14322: Function glNormalizeDegAngle( angle : Single) : Single
14323: Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14324: Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14325: Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14326: Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14327: Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14328: Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14329: Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14330: Function glArcCos( const X : Extended) : Extended;
14331: Function glArcCos1( const x : Single) : Single;
14332: Function glArcSin( const X : Extended) : Extended;
14333: Function glArcSin1( const X : Single) : Single;
14334: Function glArcTan21( const Y, X : Extended) : Extended;
14335: Function glArcTan21( const Y, X : Single) : Single;
14336: Function glFastArcTan2( y, x : Single) : Single
14337: Function glTan( const X : Extended) : Extended;
14338: Function glTan1( const X : Single) : Single;
14339: Function glCoTan( const X : Extended) : Extended;
14340: Function glCoTan1( const X : Single) : Single;
14341: Function glSinh( const x : Single) : Single;
14342: Function glSinh1( const x : Double) : Double;
14343: Function glCosh( const x : Single) : Single;
14344: Function glCosh1( const x : Double) : Double;
14345: Function glRSqrt( v : Single) : Single
14346: Function glRLength( x, y : Single) : Single
14347: Function glISqrt( i : Integer) : Integer
14348: Function glILength( x, y : Integer) : Integer;
14349: Function glILength1( x, y, z : Integer) : Integer;
14350: Procedure glRegisterBasedExp
14351: Procedure glRandomPointOnSphere( var p : TAffineVector)
14352: Function glRoundInt( v : Single) : Single;
14353: Function glRoundInt1( v : Extended) : Extended;
14354: Function glTrunc( v : Single) : Integer;
14355: Function glTrunc64( v : Extended) : Int64;
14356: Function glInt( v : Single) : Single;
14357: Function glInt1( v : Extended) : Extended;
14358: Function glFrac( v : Single) : Single;
14359: Function glFrac1( v : Extended) : Extended;
14360: Function glRound( v : Single) : Integer;
14361: Function glRound64( v : Single) : Int64;
14362: Function glRound641( v : Extended) : Int64;
14363: Function glTrunc( X : Extended) : Int64
14364: Function glRound( X : Extended) : Int64
14365: Function glFrac( X : Extended) : Extended
14366: Function glCeil( v : Single) : Integer;
14367: Function glCeil64( v : Extended) : Int64;
14368: Function glFloor( v : Single) : Integer;
14369: Function glFloor64( v : Extended) : Int64;
14370: Function glScaleAndRound( i : Integer; var s : Single) : Integer
14371: Function glSign( x : Single) : Integer
14372: Function glIsInRange( const x, a, b : Single) : Boolean;
14373: Function glIsInRangel( const x, a, b : Double) : Boolean;
14374: Function glIsInCube( const p, d : TAffineVector) : Boolean;
14375: Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14376: //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14377: //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14378: //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14379: Function glMinFloat3( const v1, v2 : Single) : Single;
14380: Function glMinFloat4( const v : array of Single) : Single;
14381: Function glMinFloat5( const v1, v2 : Double) : Double;
14382: Function glMinFloat6( const v1, v2 : Extended) : Extended;
14383: Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14384: Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14385: Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14386: //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14387: //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14388: //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14389: Function glMaxFloat2( const v : array of Single) : Single;
14390: Function glMaxFloat3( const v1, v2 : Single) : Single;
14391: Function glMaxFloat4( const v1, v2 : Double) : Double;
14392: Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14393: Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14394: Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14395: Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14396: Function glMinInteger9( const v1, v2 : Integer) : Integer;

```

```

14397: Function glMinInteger( const v1, v2 : Cardinal ) : Cardinal;
14398: Function glMaxInteger( const v1, v2 : Integer ) : Integer;
14399: Function glMaxInteger1( const v1, v2 : Cardinal ) : Cardinal;
14400: Function glTriangleArea( const p1, p2, p3 : TAffineVector ) : Single;
14401: //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer ) : Single;
14402: Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector ) : Single;
14403: //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer ) : Single;
14404: //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14405: Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14406: //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14407: Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14408: //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14409: Function glMaxXYZComponent( const v : TVectorGL ) : Single;
14410: Function glMaxXYZComponent1( const v : TAffineVector ) : Single;
14411: Function glMinXYZComponent( const v : TVectorGL ) : Single;
14412: Function glMinXYZComponent1( const v : TAffineVector ) : Single;
14413: Function glMaxAbsXYZComponent( v : TVectorGL ) : Single;
14414: Function glMinAbsXYZComponent( v : TVectorGL ) : Single;
14415: Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14416: Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14417: Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14418: Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14419: Procedure glSortArrayAscending( var a : array of Extended);
14420: Function glClampValue( const aValue, aMin, aMax : Single ) : Single;
14421: Function glClampValue1( const aValue, aMin : Single ) : Single;
14422: Function glGeometryOptimizationMode : String;
14423: Procedure glBeginFPUOnlySection;
14424: Procedure glEndFPUOnlySection;
14425: Function glConvertRotation( const Angles : TAffineVector ) : TVectorGL;
14426: Function glMakeAffineDblVector( var v : array of Double ) : TAffineDblVector;
14427: Function glMakeDblVector( var v : array of Double ) : THomogeneousDblVector;
14428: Function glVectorAffineDblToFlt( const v : TAffineDblVector ) : TAffineVector;
14429: Function glVectorDblToFlt( const v : THomogeneousDblVector ) : THomogeneousVector;
14430: Function glVectorAffineFltToDbl( const v : TAffineVector ) : TAffineDblVector;
14431: Function glVectorFltToDbl( const v : TVectorGL ) : THomogeneousDblVector;
14432: Function glPointInPolygon( var xp, yp : array of Single; x, y : Single ) : Boolean;
14433: Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word);
14434: Function glTurn( const Matrix : TMatrixGL; angle : Single ) : TMatrixGL;
14435: Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14436: Function glPitch( const Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
14437: Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14438: Function glRoll( const Matrix: TMatrixGL; Angle : Single ) : TMatrixGL;
14439: Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14440: Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single;
14441: Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
sphereCenter:TVectorGL;const sphereRadius : Single ) : Boolean;
14442: Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
const sphereRadius : Single; var i1, i2 : TVectorGL ) : Integer;
14443: Function glSphereVisibleRadius( distance, radius : Single ) : Single;
14444: Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL ) : TFrustum;
14445: Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
TRenderContextClippingInfo ) : Boolean;
14446: Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
TRenderContextClippingInfo ) : Boolean;
14447: Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14448: Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
Frustum:TFrustum):Bool;
14449: Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL ) : TMatrixGL;
14450: Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL ) : TMatrixGL;
14451: Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector ) : TMatrixGL;
14452: Function glPackRotationMatrix( const mat : TMatrixGL ) : TPackedRotationMatrix;
14453: Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix ) : TMatrixGL;
14454: 'cPI','Single').setExtended( 3.141592654);
14455: 'cPIdiv180','Single').setExtended( 0.017453292);
14456: 'c180divPI','Single').setExtended( 57.29577951);
14457: 'c2PI','Single').setExtended( 6.283185307);
14458: 'cPIdiv2','Single').setExtended( 1.570796326);
14459: 'cPIdiv4','Single').setExtended( 0.785398163);
14460: 'c3PIdiv4','Single').setExtended( 2.35619449);
14461: 'cInv2PI','Single').setExtended( 1 / 6.283185307);
14462: 'cInv360','Single').setExtended( 1 / 360);
14463: 'c180','Single').setExtended( 180);
14464: 'c360','Single').setExtended( 360);
14465: 'cOneHalf','Single').setExtended( 0.5);
14466: 'cLn10','Single').setExtended( 2.302585093);
14467: {'MinSingle','Extended').setExtended( 1.5e-45);
14468: 'MaxSingle','Extended').setExtended( 3.4e+38);
14469: 'MinDouble','Extended').setExtended( 5.0e-324);
14470: 'MaxDouble','Extended').setExtended( 1.7e+308);
14471: 'MinExtended','Extended').setExtended( 3.4e-4932);
14472: 'MaxExtended','Extended').setExtended( 1.1e+4932);
14473: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14474: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14475: end;
14476:
14477: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14478: begin
14479:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14480:   (FindClass('TOBJECT'),'TFaceGroups

```



```

14481: TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14482: TMeshAutoCenterings', 'set of TMeshAutoCentering
14483: TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14484: SIRegister_TBBaseMeshObject(CL);
14485: (FindClass('TOBJECT'), 'TSkeletonFrameList
14486: TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14487: SIRegister_TSkeletonFrame(CL);
14488: SIRegister_TSkeletonFrameList(CL);
14489: (FindClass('TOBJECT'), 'TSkeleton
14490: (FindClass('TOBJECT'), 'TSkeletonBone
14491: SIRegister_TSkeletonBoneList(CL);
14492: SIRegister_TSkeletonRootBoneList(CL);
14493: SIRegister_TSkeletonBone(CL);
14494: (FindClass('TOBJECT'), 'TSkeletonColliderList
14495: SIRegister_TSkeletonCollider(CL);
14496: SIRegister_TSkeletonColliderList(CL);
14497: (FindClass('TOBJECT'), 'TGLBaseMesh
14498: TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14499:   + 'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14500:   + 'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14501:   + 'QuaternionList; end
14502: SIRegister_TSkeleton(CL);
14503: TMeshObjectRenderingOption', '( moroGroupByMaterial )
14504: TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14505: SIRegister_TMeshObject(CL);
14506: SIRegister_TMeshObjectList(CL);
14507: //TMeshObjectListClass', 'class of TMeshObjectList
14508: (FindClass('TOBJECT'), 'TMeshMorphTargetList
14509: SIRegister_TMeshMorphTarget(CL);
14510: SIRegister_TMeshMorphTargetList(CL);
14511: SIRegister_TMorphableMeshObject(CL);
14512: TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14513: //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not work
14514: //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14515: TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14516: SIRegister_TSkeletonMeshObject(CL);
14517: SIRegister_TFaceGroup(CL);
14518: TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14519:   + 'atTriangles, fgmmTriangleFan, fgmmQuads )
14520: SIRegister_TFGVertexIndexList(CL);
14521: SIRegister_TFGVertexNormalTexIndexList(CL);
14522: SIRegister_TFGIndexTexCoordList(CL);
14523: SIRegister_TFaceGroups(CL);
14524: TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14525: SIRegister_TVectorFile(CL);
14526: //TVectorFileClass', 'class of TVectorFile
14527: SIRegister_TGLGLSMVectorFile(CL);
14528: SIRegister_TGLBaseMesh(CL);
14529: SIRegister_TGLFreeForm(CL);
14530: TGLActorOption', '( aoSkeletonNormalizeNormals )
14531: TGLActorOptions', 'set of TGLActorOption
14532: 'cDefaultGLActorOptions', 'LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14533: (FindClass('TOBJECT'), 'TGLActor
14534: TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14535: SIRegister_TActorAnimation(CL);
14536: TActorAnimationName', 'String
14537: SIRegister_TActorAnimations(CL);
14538: SIRegister_TGLBaseAnimationControler(CL);
14539: SIRegister_TGLAnimationControler(CL);
14540: TActorFrameInterpolation', '( afpNone, afpLinear )
14541: TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14542:   + 'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14543: SIRegister_TGLActor(CL);
14544: SIRegister_TVectorFileFormat(CL);
14545: SIRegister_TVectorFileFormatsList(CL);
14546: (FindClass('TOBJECT'), 'EInvalidVectorFile
14547: Function GetVectorFileFormats : TVectorFileFormatsList
14548: Function VectorFileFormatsFilter : String
14549: Function VectorFileFormatsSaveFilter : String
14550: Function VectorFileFormatExtensionByIndex( index : Integer) : String
14551: Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14552: Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14553: end;
14554:
14555: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14556: begin
14557:   'Class_DColorPropPage', 'TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}'
14558:   'Class_DFontPropPage', 'TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}'
14559:   'Class_DPicturePropPage', 'TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}'
14560:   'Class_DStringPropPage', 'TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}'
14561:   SIRegister_TOLEStream(CL);
14562:   (FindClass('TOBJECT'), 'TConnectionPoints
14563:   TConnectionKind', '( ckSingle, ckMulti )
14564:   SIRegister_TConnectionPoint(CL);
14565:   SIRegister_TConnectionPoints(CL);
14566:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14567:   (FindClass('TOBJECT'), 'TActiveXControlFactory
14568:   SIRegister_TActiveXControl(CL);
14569:   //TActiveXControlClass', 'class of TActiveXControl

```

```

14570: SIRegister_TActiveXControlFactory(CL);
14571: SIRegister_TActiveFormControl(CL);
14572: SIRegister_TActiveForm(CL);
14573: //TActiveFormClass', 'class of TActiveForm
14574: SIRegister_TActiveFormFactory(CL);
14575: (FindClass('TOBJECT'), 'TPropertyPageImpl
14576: SIRegister_TPropertyPage(CL);
14577: //TPropertyPageClass', 'class of TPropertyPage
14578: SIRegister_TPropertyPageImpl(CL);
14579: SIRegister_TActiveXPropertyPage(CL);
14580: SIRegister_TActiveXPropertyPageFactory(CL);
14581: SIRegister_TCustomAdapter(CL);
14582: SIRegister_TAdapterNotifier(CL);
14583: SIRegister_IFontAccess(CL);
14584: SIRegister_TFontAdapter(CL);
14585: SIRegister_IPictureAccess(CL);
14586: SIRegister_TPictureAdapter(CL);
14587: SIRegister_TOLEGraphic(CL);
14588: SIRegister_TStringsAdapter(CL);
14589: SIRegister_TReflectorWindow(CL);
14590: Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14591: Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14592: Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14593: Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14594: Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14595: Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14596: Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14597: Function ParkingWindow : HWND
14598: end;
14599:
14600: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14601: begin
14602: // TIp6Bytes = array [0..15] of Byte;
14603: {:binary form of IPv6 adress (for string conversion routines)}
14604: // TIp6Words = array [0..7] of Word;
14605: AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14606: AddTypeS('TIp6Words', 'array [0..7] of Word;');
14607: AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end;');
14608: Function synaIsIP( const Value : string) : Boolean;
14609: Function synaIsIP6( const Value : string) : Boolean;
14610: Function synaIPToID( Host : string) : AnsiString;
14611: Function synaStrToIp6( value : string) : TIp6Bytes;
14612: Function synaIp6ToStr( value : TIp6Bytes) : string;
14613: Function synaStrToIp( value : string) : integer;
14614: Function synaIpToStr( value : integer) : string;
14615: Function synaReverseIP( Value : AnsiString) : AnsiString;
14616: Function synaReverseIP6( Value : AnsiString) : AnsiString;
14617: Function synaExpandIP6( Value : AnsiString) : AnsiString;
14618: Function xStrToIP( const Value : String) : TIPAdr;
14619: Function xIPToStr( const Adresse : TIPAdr) : String;
14620: Function IPToCardinal( const Adresse : TIPAdr) : Cardinal;
14621: Function CardinalToIP( const Value : Cardinal) : TIPAdr;
14622: end;
14623:
14624: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14625: begin
14626: AddTypeS('TSpecials', 'set of Char');
14627: Const('SpecialChar', 'TSpecials').SetSet( '=[<>:;,@/?\`_']');
14628: Const('URLFullSpecialChar', 'TSpecials').SetSet( ';/?:@=&#%+');
14629: Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/='');
14630: Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/,='');
14631: Const('TableUU'('!'#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNopqrstuvwxyz[\]^_');
14632: Const('TableXX'(+0123456789ABCDEFGHIJKLMNopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ);
14633: Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString;
14634: Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString;
14635: Function DecodeURL( const Value : AnsiString) : AnsiString;
14636: Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString;
14637: Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString;
14638: Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString;
14639: Function EncodeURLElement( const Value : AnsiString) : AnsiString;
14640: Function EncodeURL( const Value : AnsiString) : AnsiString;
14641: Function Decode4to3( const Value, Table : AnsiString) : AnsiString;
14642: Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString;
14643: Function Encode3to4( const Value, Table : AnsiString) : AnsiString;
14644: Function synDecodeBase64( const Value : AnsiString) : AnsiString;
14645: Function synEncodeBase64( const Value : AnsiString) : AnsiString;
14646: Function DecodeBase64mod( const Value : AnsiString) : AnsiString;
14647: Function EncodeBase64mod( const Value : AnsiString) : AnsiString;
14648: Function DecodeUU( const Value : AnsiString) : AnsiString;
14649: Function EncodeUU( const Value : AnsiString) : AnsiString;
14650: Function DecodeXX( const Value : AnsiString) : AnsiString;
14651: Function DecodeYEnc( const Value : AnsiString) : AnsiString;
14652: Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer;
14653: Function synCrc32( const Value : AnsiString) : Integer;
14654: Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word;
14655: Function Crc16( const Value : AnsiString) : Word;
14656: Function synMD5( const Value : AnsiString) : AnsiString;
14657: Function HMAC_MD5( Text, Key : AnsiString) : AnsiString;
14658: Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString;

```

```

14659: Function synSHA1( const Value : AnsiString) : AnsiString);
14660: Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString);
14661: Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString);
14662: Function synMD4( const Value : AnsiString) : AnsiString);
14663: end;
14664:
14665: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14666: begin
14667:   AddTypeS('TTimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14668:   + 'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14669:   + 'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14670:   + '4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14671:   + '_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14672:   + 'C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14673:   + ', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14674:   + ', NEXTSTEP, ARMSCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14675:   + 'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14676:   + '1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14677:   + '2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN,'
14678:   + 'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14679:   + 'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14680:   + ', CP864, CP865, CP869, CP1125 )');
14681:   AddTypeS('TTimeSetChar', 'set of TTimeChar');
14682:   Function CharsetConversion(const Value:AnsiString;CharFrom:TTimeChar;CharTo:TTimeChar):AnsiString;
14683:   Function CharsetConversionEx(const Value:AnsiString;CharFrom:TTimeChar;CharTo:TTimeChar; const
TransformTable : array of Word) : AnsiString);
14684:   Function CharsetConversionTrans( Value : AnsiString; CharFrom : TTimeChar; CharTo : TTimeChar; const
TransformTable : array of Word; Translit : Boolean) : AnsiString);
14685:   Function GetCurCP : TTimeChar);
14686:   Function GetCurOEMCP : TTimeChar);
14687:   Function GetCPFromID( Value : AnsiString) : TTimeChar);
14688:   Function GetIDFromCP( Value : TTimeChar) : AnsiString);
14689:   Function NeedCharsetConversion( const Value : AnsiString) : Boolean);
14690:   Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TTimeChar;CharTo:TTimeSetChar):TTimeChar;
14691:   Function GetBOM( Value : TTimeChar) : AnsiString);
14692:   Function StringToWide( const Value : AnsiString) : WideString);
14693:   Function WideToString( const Value : WideString) : AnsiString);
14694: end;
14695:
14696: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14697: begin
14698:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14699:   Procedure WakeOnLan( MAC, IP : string);
14700:   Function GetDNS : string);
14701:   Function GetIEProxy( protocol : string) : TProxySetting);
14702:   Function GetLocalIPs : string);
14703: end;
14704:
14705:
14706: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14707: begin
14708:   AddConstantN('synCR', 'Char #0d);
14709:   Const('synLF', 'Char #0a);
14710:   Const('cSerialChunk', 'LongInt'( 8192);
14711:   Const('LockfileDirectory', 'String '/var/lock');
14712:   Const('PortIsClosed', 'LongInt'( - 1);
14713:   Const('ErrAlreadyOwned', 'LongInt'( 9991);
14714:   Const('ErrAlreadyInUse', 'LongInt'( 9992);
14715:   Const('ErrWrongParameter', 'LongInt'( 9993);
14716:   Const('ErrPortNotOpen', 'LongInt'( 9994);
14717:   Const('ErrNoDeviceAnswer', 'LongInt'( 9995);
14718:   Const('ErrMaxBuffer', 'LongInt'( 9996);
14719:   Const('ErrTimeout', 'LongInt'( 9997);
14720:   Const('ErrNotRead', 'LongInt'( 9998);
14721:   Const('ErrFrame', 'LongInt'( 9999);
14722:   Const('ErrOverrun', 'LongInt'( 10000);
14723:   Const('ErrRxOver', 'LongInt'( 10001);
14724:   Const('ErrRxParity', 'LongInt'( 10002);
14725:   Const('ErrTxFull', 'LongInt'( 10003);
14726:   Const('dcb_Binary', 'LongWord')( $00000001);
14727:   Const('dcb_ParityCheck', 'LongWord')( $00000002);
14728:   Const('dcb_OutxCtsFlow', 'LongWord')( $00000004);
14729:   Const('dcb_OutxDsrFlow', 'LongWord')( $00000008);
14730:   Const('dcb_DtrControlMask', 'LongWord')( $00000030);
14731:   Const('dcb_DtrControlDisable', 'LongWord')( $00000000);
14732:   Const('dcb_DtrControlEnable', 'LongWord')( $00000010);
14733:   Const('dcb_DtrControlHandshake', 'LongWord')( $00000020);
14734:   Const('dcb_DsrSensitivity', 'LongWord')( $00000040);
14735:   Const('dcb_TXContinueOnXoff', 'LongWord')( $00000080);
14736:   Const('dcb_OutX', 'LongWord')( $00000100);
14737:   Const('dcb_InX', 'LongWord')( $00000200);
14738:   Const('dcb_ErrorChar', 'LongWord')( $00000400);
14739:   Const('dcb_NullStrip', 'LongWord')( $00000800);
14740:   Const('dcb_RtsControlMask', 'LongWord')( $00003000);
14741:   Const('dcb_RtsControlDisable', 'LongWord')( $00000000);
14742:   Const('dcb_RtsControlEnable', 'LongWord')( $00001000);
14743:   Const('dcb_RtsControlHandshake', 'LongWord')( $00002000);
14744:   Const('dcb_RtsControlToggle', 'LongWord')( $00003000);
14745:   Const('dcb_AbortOnError', 'LongWord')( $00004000);

```

```

14746: Const('dcb_Reserveds','LongWord')($FFFF8000);
14747: Const('synSB1','LongInt'( 0));
14748: Const('SBlandHalf','LongInt'( 1));
14749: Const('synSB2','LongInt'( 2));
14750: Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14751: Const('CS7fix','LongWord')($0000020);
14752: AddTypes('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14753:   +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14754:   +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14755:   +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14756: //AddTypes('PDCB', '^TDCB // will not work');
14757: //Const('MaxRates','LongInt'( 18);
14758: //Const('MaxRates','LongInt'( 30);
14759: //Const('MaxRates','LongInt'( 19);
14760: Const('O_SYNC','LongWord')($0080);
14761: Const('synOK','LongInt'( 0);
14762: Const('synErr','LongInt'( integer ( - 1 ));
14763: AddTypes('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
HR_WriteCount, HR_Wait )');
14764: Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string));
14765: SIRegister_ESynaSerError(CL);
14766: SIRegister_TBlockSerial(CL);
14767: Function GetSerialPortNames : string);
14768: end;
14769:
14770: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14771: begin
14772:   Const('DLLIconvName','String 'libiconv.so');
14773:   Const('DLLIconvName','String 'iconv.dll');
14774:   AddTypes('size_t', 'Cardinal');
14775:   AddTypes('iconv_t', 'Integer');
14776:   //AddTypes('iconv_t', 'Pointer');
14777:   AddTypes('argptr', 'iconv_t');
14778:   Function SynaIconvOpen( const tocode, fromcode : AnsiString) : iconv_t);
14779:   Function SynaIconvOpenTranslit( const tocode, fromcode : AnsiString) : iconv_t);
14780:   Function SynaIconvOpenIgnore( const tocode, fromcode : AnsiString) : iconv_t);
14781:   Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer);
14782:   Function SynaIconvClose( var cd : iconv_t) : integer);
14783:   Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer);
14784:   Function IsIconvloaded : Boolean);
14785:   Function InitIconvInterface : Boolean);
14786:   Function DestroyIconvInterface : Boolean);
14787:   Const('ICONV_TRIVIALP','LongInt'( 0);
14788:   Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14789:   Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14790:   Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14791:   Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14792: end;
14793:
14794: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14795: begin
14796:   Const 'ICMP_ECHO','LongInt'( 8);
14797:   Const('ICMP_ECHOREPLY','LongInt'( 0);
14798:   Const('ICMP_UNREACH','LongInt'( 3);
14799:   Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14800:   Const('ICMP6_ECHO','LongInt'( 128);
14801:   Const('ICMP6_ECHOREPLY','LongInt'( 129);
14802:   Const('ICMP6_UNREACH','LongInt'( 1);
14803:   Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14804:   AddTypes('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14805:     +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14806:   SIRegister_TPINGSend(CL);
14807:   Function PingHost( const Host : string) : Integer);
14808:   Function TraceRouteHost( const Host : string) : string);
14809: end;
14810:
14811: procedure SIRegister_asnlutil(CL: TPSPascalCompiler);
14812: begin
14813:   AddConstantN('synASN1_BOOL','LongWord')($01);
14814:   Const('synASN1_INT','LongWord')($02);
14815:   Const('synASN1_OCTSTR','LongWord')($04);
14816:   Const('synASN1_NULL','LongWord')($05);
14817:   Const('synASN1_OBJID','LongWord')($06);
14818:   Const('synASN1_ENUM','LongWord')($0a);
14819:   Const('synASN1_SEQ','LongWord')($30);
14820:   Const('synASN1_SETOF','LongWord')($31);
14821:   Const('synASN1_IPADDR','LongWord')($40);
14822:   Const('synASN1_COUNTER','LongWord')($41);
14823:   Const('synASN1_GAUGE','LongWord')($42);
14824:   Const('synASN1_TIMETICKS','LongWord')($43);
14825:   Const('synASN1_OPAQUE','LongWord')($44);
14826:   Function synASNEncOIDItem( Value : Integer) : AnsiString);
14827:   Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer);
14828:   Function synASNEncLen( Len : Integer) : AnsiString);
14829:   Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer);
14830:   Function synASNEncInt( Value : Integer) : AnsiString);
14831:   Function synASNEncUInt( Value : Integer) : AnsiString);
14832:   Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString);
14833:   Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;

```



```

14834: Function synMibToId( Mib : String ) : AnsiString';
14835: Function synIdToMib( const Id : AnsiString ) : String';
14836: Function synIntMibToStr( const Value : AnsiString ) : AnsiString';
14837: Function ASNDump( const Value : AnsiString ) : AnsiString';
14838: Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean';
14839: Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString';
14840: end;
14841:
14842: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14843: begin
14844:   Const('cLDAPProtocol','String '389');
14845:   Const('LDAP_ASN1_BIND_REQUEST','LongWord')($60);
14846:   Const('LDAP_ASN1_BIND_RESPONSE','LongWord')($61);
14847:   Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')($42);
14848:   Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')($63);
14849:   Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')($64);
14850:   Const('LDAP_ASN1_SEARCH_DONE','LongWord')($65);
14851:   Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')($73);
14852:   Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')($66);
14853:   Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')($67);
14854:   Const('LDAP_ASN1_ADD_REQUEST','LongWord')($68);
14855:   Const('LDAP_ASN1_ADD_RESPONSE','LongWord')($69);
14856:   Const('LDAP_ASN1_DEL_REQUEST','LongWord')($4A);
14857:   Const('LDAP_ASN1_DEL_RESPONSE','LongWord')($6B);
14858:   Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')($6C);
14859:   Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')($6D);
14860:   Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')($6E);
14861:   Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')($6F);
14862:   Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')($70);
14863:   Const('LDAP_ASN1_EXT_REQUEST','LongWord')($77);
14864:   Const('LDAP_ASN1_EXT_RESPONSE','LongWord')($78);
14865:   SIRegister_TLDAPAttribute(CL);
14866:   SIRegister_TLDAPAttributeList(CL);
14867:   SIRegister_TLDAPResult(CL);
14868:   SIRegister_TLDAPResultList(CL);
14869:   AddTypeS('TLDAPOModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14870:   AddTypeS('LDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14871:   AddTypeS('LDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14872:   SIRegister_TLDAPSend(CL);
14873:   Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString';
14874: end;
14875:
14876:
14877: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14878: begin
14879:   Const('cSysLogProtocol','String '514');
14880:   Const('FCL_Kernel','LongInt'( 0);
14881:   Const('FCL_UserLevel','LongInt'( 1);
14882:   Const('FCL_MailSystem','LongInt'( 2);
14883:   Const('FCL_System','LongInt'( 3);
14884:   Const('FCL_Security','LongInt'( 4);
14885:   Const('FCL_Syslogd','LongInt'( 5);
14886:   Const('FCL_Printer','LongInt'( 6);
14887:   Const('FCL_News','LongInt'( 7);
14888:   Const('FCL_UUCP','LongInt'( 8);
14889:   Const('FCL_Clock','LongInt'( 9);
14890:   Const('FCL_Authorization','LongInt'( 10);
14891:   Const('FCL_FTP','LongInt'( 11);
14892:   Const('FCL_NTP','LongInt'( 12);
14893:   Const('FCL_LogAudit','LongInt'( 13);
14894:   Const('FCL_LogAlert','LongInt'( 14);
14895:   Const('FCL_Time','LongInt'( 15);
14896:   Const('FCL_Local0','LongInt'( 16);
14897:   Const('FCL_Local1','LongInt'( 17);
14898:   Const('FCL_Local2','LongInt'( 18);
14899:   Const('FCL_Local3','LongInt'( 19);
14900:   Const('FCL_Local4','LongInt'( 20);
14901:   Const('FCL_Local5','LongInt'( 21);
14902:   Const('FCL_Local6','LongInt'( 22);
14903:   Const('FCL_Local7','LongInt'( 23);
14904:   AddTypeS('TSyslogSeverity', '( Emergency, Alert, Critical, Error, Warning,
14905:   + ' Notice, Info, Debug )');
14906:   SIRegister_TSyslogMessage(CL);
14907:   SIRegister_TSyslogSend(CL);
14908:   Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
Content:string): Boolean;
14909: end;
14910:
14911:
14912: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14913: begin
14914:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14915:   SIRegister_TMessHeader(CL);
14916:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14917:   SIRegister_TMimeMess(CL);
14918: end;
14919:
14920: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14921: begin

```

```

14922: (FindClass('TOBJECT'),'TMimePart');
14923: AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
14924: AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14925: AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14926: SIRegister_TMimePart(CL);
14927: Const('MaxMimeType','LongInt'( 25);
14928: Function GenerateBoundary : string');
14929: end;
14930:
14931: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
14932: begin
14933:   Function InlineDecode( const Value : string; CP : TMimeChar ) : string');
14934:   Function InlineEncode( const Value : string; CP, MimeP : TMimeChar ) : string');
14935:   Function NeedInline( const Value : AnsiString ) : boolean');
14936:   Function InlineCodeEx( const Value : string; FromCP : TMimeChar ) : string');
14937:   Function InlineCode( const Value : string ) : string');
14938:   Function InlineEmailEx( const Value : string; FromCP : TMimeChar ) : string');
14939:   Function InlineEmail( const Value : string ) : string');
14940: end;
14941:
14942: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
14943: begin
14944:   Const('cFtpProtocol','String '21');
14945:   Const('cFtpDataProtocol','String '20');
14946:   Const('FTP_OK','LongInt'( 255);
14947:   Const('FTP_ERR','LongInt'( 254);
14948:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
14949:   SIRegister_TFTPListRec(CL);
14950:   SIRegister_TFTPList(CL);
14951:   SIRegister_TFTPSend(CL);
14952:   Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean');
14953:   Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean');
14954:   Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
ToPort, ToFile, ToUser, ToPass : string ) : Boolean');
14955: end;
14956:
14957: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
14958: begin
14959:   Const('cHttpProtocol','String '80');
14960:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
14961:   SIRegister_THTTPSend(CL);
14962:   Function HttpGetText( const URL : string; const Response : TStrings ) : Boolean');
14963:   Function HttpGetBinary( const URL : string; const Response : TStream ) : Boolean');
14964:   Function HttpPostBinary( const URL : string; const Data : TStream ) : Boolean');
14965:   Function HttpPostURL( const URL, URLData : string; const Data : TStream ) : Boolean');
14966:   Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
ResultData:TStrings):Bool;
14967: end;
14968:
14969: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
14970: begin
14971:   Const('cSmtpprotocol','String '25');
14972:   SIRegister_TSMTPSend(CL);
14973:   Function SendToRaw(const MailFrom,MailTo,SMTPHost:string;const MailData:TStrings;const Username,
Passw:string):Bool;
14974:   Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
14975:   Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
Username, Password : string):Boolean');
14976: end;
14977:
14978: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
14979: begin
14980:   Const('cSnmpProtocol','String '161);
14981:   Const('cSnmpTrapProtocol','String '162);
14982:   Const('SNMP_V1','LongInt'( 0);
14983:   Const('SNMP_V2C','LongInt'( 1);
14984:   Const('SNMP_V3','LongInt'( 3);
14985:   Const('PDUGetRequest','LongWord')($A0);
14986:   Const('PDUGetNextRequest','LongWord')($A1);
14987:   Const('PDUGetResponse','LongWord')($A2);
14988:   Const('PDUSetRequest','LongWord')($A3);
14989:   Const('PDUTrap','LongWord')($A4);
14990:   Const('PDUGetBulkRequest','LongWord')($A5);
14991:   Const('PDUInformRequest','LongWord')($A6);
14992:   Const('PDUTrapV2','LongWord')($A7);
14993:   Const('PDUREport','LongWord')($A8);
14994:   Const('ENoError','LongInt' 0;
14995:   Const('ETooBig','LongInt')( 1);
14996:   Const('ENoSuchName','LongInt'( 2);
14997:   Const('EBadValue','LongInt'( 3);
14998:   Const('EReadOnly','LongInt'( 4);
14999:   Const('EGenErr','LongInt'( 5);
15000:   Const('ENoAccess','LongInt'( 6);
15001:   Const('EWrongType','LongInt'( 7);
15002:   Const('EWrongLength','LongInt'( 8);
15003:   Const('EWrongEncoding','LongInt'( 9);
15004:   Const('EWrongValue','LongInt'( 10);
15005:   Const('ENoCreation','LongInt'( 11);
15006:   Const('EInconsistentValue','LongInt'( 12);

```

```

15007: Const('EResourceUnavailable','LongInt'( 13);
15008: Const('ECommitFailed','LongInt'( 14);
15009: Const('EUndoFailed','LongInt'( 15);
15010: Const('EAuthorizationError','LongInt'( 16);
15011: Const('ENotWritable','LongInt'( 17);
15012: Const('EInconsistentName','LongInt'( 18);
15013: AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15014: AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15015: AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15016: SIRegister_TSNMPMib(CL);
15017: AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15018:   +'EngineTime : integer; EngineStamp : Cardinal; end');
15019: SIRegister_TSNMPRec(CL);
15020: SIRegister_TSNMPSend(CL);
15021: Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean);
15022: Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean);
15023: Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15024: Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15025: Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):
Boolean;
15026: Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
Integer; const MIBName, MIBValue : AnsiString; MIBType : Integer) : Integer);
15027: Function RecvTrap( var Dest, Source, Enterprise, Community : AnsiString; var Generic, Specific, Seconds :
Integer; const MIBName, MIBValue : TStringList) : Integer);
15028: end;
15029:
15030: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15031: begin
15032:   Function GetDomainName2: AnsiString);
15033:   Function GetDomainController( Domain : AnsiString) : AnsiString);
15034:   Function GetDomainUsers( Controller : AnsiString) : AnsiString);
15035:   Function GetDomainGroups( Controller : AnsiString) : AnsiString);
15036:   Function GetDateTime( Controller : AnsiString) : TDateTime);
15037:   Function GetFullName2( Controller, UserID : AnsiString) : AnsiString);
15038: end;
15039:
15040: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15041: begin
15042:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15043:   TwwDateTimeSelection', '(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15044:   Function wwStrToDate( const S : string) : boolean);
15045:   Function wwStrToTime( const S : string) : boolean);
15046:   Function wwStrToDateTime( const S : string) : boolean);
15047:   Function wwStrToTimeVal( const S : string) : TDateTime);
15048:   Function wwStrToDateVal( const S : string) : TDateTime);
15049:   Function wwStrToDateTimeVal( const S : string) : TDateTime);
15050:   Function wwStrToInt( const S : string) : boolean);
15051:   Function wwStrToFloat( const S : string) : boolean);
15052:   Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder);
15053:   Function wwNextDay( Year, Month, Day : Word) : integer);
15054:   Function wwPriorDay( Year, Month, Day : Word) : integer);
15055:   Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean);
15056:   Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean);
15057:   Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection);
15058:   Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection);
15059:   Function wwScanDate( const S : string; var Date : TDateTime) : Boolean);
15060:   Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean);
15061:   Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15062:   Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat : string) : boolean);
15063: end;
15064:
15065: unit uPSI_Themes;
15066: Function ThemeServices : TThemeServices);
15067: Function ThemeControl( AControl : TControl) : Boolean);
15068: Function UnthemedDesigner( AControl : TControl) : Boolean);
15069: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15070: begin
15071:   Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String);
15072: end;
15073: Unit uPSC_menus;
15074: Function StripHotkey( const Text : string) : string);
15075: Function GetHotkey( const Text : string) : string);
15076: Function AnsiSameCaption( const Text1, Text2 : string) : Boolean);
15077: Function IsAltGRPressed : boolean);
15078:
15079: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15080: begin
15081:   TCommandEvent', 'Procedure(Thread:TidPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15082:   SIRegister_TidIMAP4Server(CL);
15083: end;
15084:
15085: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15086: begin
15087:   'HASH_SIZE','LongInt'( 256);
15088:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15089:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15090:   //CL.AddTypeS('PSymbol', 'TSymbol // will not work');
15091:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15092:   + ' : Integer; Value : Variant; end');

```

```

15093: //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15094: SIRegister_TVARIANTSymbolTable(CL);
15095: end;
15096:
15097: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15098: begin
15099:   SIRegister_TThreadLocalVariables(CL);
15100:   Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD ) : PChar;
15101:   //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD ) : PISC_QUAD;
15102:   Function ThreadLocals : TThreadLocalVariables;
15103:   Procedure WriteDebug( sz : String );
15104:   CL.AddConstantN('UDF_SUCCESS','LongInt'( 0 );
15105:   'UDF_FAILURE','LongInt'( 1 );
15106:   'cSignificantlyLarger','LongInt'( 1024 * 4 );
15107:   CL.AddTypeS('mTByteArray', 'array of byte;');
15108:   function ChangeOEPFFromBytes(bFile:mTByteArray):Boolean;
15109:   function ChangeOEPFFromFile(sFile:string; sDestFile:string):Boolean;
15110:   procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15111:   function IsNetworkConnected: Boolean;
15112:   function IsInternetConnected: Boolean;
15113:   function IsCOMConnected: Boolean;
15114:   function IsNetworkOn: Boolean;
15115:   function IsInternetOn: Boolean;
15116:   function IsCOMOn: Boolean;
15117:   Function SetTimer(hWnd: HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15118:   Function KillTimer( hWnd: HWND; uIDEvent : UINT ) : BOOL;
15119:   Function wIsWindowUnicode( hWnd: HWND ) : BOOL;
15120:   Function wEnableWindow( hWnd: HWND; bEnable : BOOL ) : BOOL;
15121:   Function wIsWindowEnabled( hWnd: HWND ) : BOOL;
15122:   Function GetMenu( hWnd: HWND ) : HMENU;
15123:   Function SetMenu( hWnd: HWND; hMenu : HMENU ) : BOOL;
15124: end;
15125:
15126: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15127: begin
15128:   SIRegister_IDataBlock(CL);
15129:   SIRegister_ISendDataBlock(CL);
15130:   SIRegister_ITransport(CL);
15131:   SIRegister_TDataBlock(CL);
15132:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15133:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15134:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15135:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15136:   SIRegister_TCustomDataBlockInterpreter(CL);
15137:   SIRegister_TSendDataBlock(CL);
15138:   'CallSig','LongWord')( $D800);
15139:   'ResultSig','LongWord')( $D400);
15140:   'asMask','LongWord')( $00FF);
15141:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15142:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15143:   Procedure CheckSignature( Sig : Integer );
15144: end;
15145:
15146: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15147: begin
15148:   //CL.AddTypeS('HINTERNET', '___Pointer');
15149:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15150:   CL.AddTypeS('HINTERNET', 'Integer');
15151:   CL.AddTypeS('HINTERNET2', '___Pointer');
15152:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15153:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15154:   CL.AddTypeS('INTERNET_PORT', 'Word');
15155:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15156:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15157:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15158:   'INTERNET_RFC1123_FORMAT','LongInt'( 0 );
15159:   'INTERNET_RFC1123_BUFSIZE','LongInt'( 30 );
15160:   Function InternetCrackUrl(lpszUrl:PChar; dwUrlLength,dwFlags:DWORD; var
lpUrlComponents:TURLComponents):BOOL;
15161:   Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
dwUrlLength:DWORD):BOOL;
15162:   Function InternetCloseHandle( hInet : HINTERNET ) : BOOL;
15163:   Function
InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15164:   Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
;dwContext:DWORD):HINTERNET;
15165:   Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
lpszProxBypass:PChar;dwFlags:DWORD);
15166:   Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumOfBytesAvail:DWORD;dwFlags,
dwContext:DWORD):BOOL;
15167:   Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE ) : BOOL;
15168:   Function
InternetDial(hwndParent:HWND;lpszConnect:PChar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15170:   Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD ) : DWORD;
15171:   Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD ) : BOOL;
15172:   Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD ) : BOOL;
15173:   Function InternetAutodialHangup( dwReserved : DWORD ) : BOOL;
15174:   Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD ) : BOOL;

```



```

15175: Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
    lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15176: Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
    lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15177: Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15178: ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15179: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumOfBytesAvail:DWORD;dwFlgs,
    dwContext:DWORD):BOOL;
15180: Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
    Twin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET';
15181: Function WFTpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
    BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL';
15182: Function
    WFTpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15183: Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL';
15184: Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15185: Function
    FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15186: Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL';
15187: Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL';
15188: Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL';
15189: Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15190: Function
    FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommnd:PChar;dwContxt:DWORD):BOOL;
15191: Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL';
15192: Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL';
15193: Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL';
15194: Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL';
15195: Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL';
15196: Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL';
15197: Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL';
15198: Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL';
15199: Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL';
15200: Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL';
15201: Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL';
15202: Function
    GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
    PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15203: Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL';
15204: Function
    GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15205: Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
    PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15206: Function
    HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15207: Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
    dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15208: Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15209: Function InternetAttemptConnect( dwReserved : DWORD) : DWORD';
15210: Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL';
15211: Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15212: Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD';
15213: Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64';
15214: Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool';
15215: Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
    lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15216: Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
    lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15217: Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15218: Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15219: Function
    InternetDial(hWndParent:HWND;lpszConnect:PChar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15220: Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL';
15221: end;
15222:
15223: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15224: begin
15225:     AddTypeS('strCharSet', 'set of char');
15226:     TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15227:     AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15228:     Procedure strBreakApart( s : string; delimiter : string; parts : TStrings);
15229:     Function strGetToken( s : string; delimiter : string; var APos : integer) : string';
15230:     Procedure strStripPreceding( var s : string; delimiter : strCharSet);
15231:     Procedure strStripTrailing( var s : string; delimiter : strCharSet);
15232:     Procedure strStripWhiteSpace( var s : string);
15233:     Function strRemoveChar( str : string; removeChar : char) : string';
15234:     Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string';
15235:     Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string';
15236:     Function wwEqualStr( s1, s2 : string) : boolean';
15237:     Function strCount( s : string; delimiter : char) : integer';
15238:     Function strWhiteSpace : strCharSet';
15239:     Function wwExtractFileNameOnly( const FileName : string) : string';
15240:     Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15241:     Function strTrailing( s : string; delimiter : char) : string';
15242:     Function strPreceding( s : string; delimiter : char) : string';
15243:     Function wwstrReplace( s, Find, Replace : string) : string';
15244: end;
15245:
15246: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);

```

```

15247: begin
15248:   SIRegister_TRemoteDataModule(CL);
15249:   SIRegister_TCRremoteDataModule(CL);
15250:   Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean);
15251:   Procedure UnregisterPooled( const ClassID : string);
15252:   Procedure EnableSocketTransport( const ClassID : string);
15253:   Procedure DisableSocketTransport( const ClassID : string);
15254:   Procedure EnableWebTransport( const ClassID : string);
15255:   Procedure DisableWebTransport( const ClassID : string);
15256: end;
15257:
15258: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15259: begin
15260:   Function mxArcCos( x : Real ) : Real;
15261:   Function mxArcSin( x : Real ) : Real;
15262:   Function Comp2Str( N : Comp ) : String;
15263:   Function Int2StrPad0( N : LongInt; Len : Integer ) : String;
15264:   Function Int2Str( N : LongInt ) : String;
15265:   Function mxIsEqual( R1, R2 : Double ) : Boolean;
15266:   Function LogXY( x, y : Real ) : Real;
15267:   Function Pennies2Dollars( C : Comp ) : String;
15268:   Function mxPower( X : Integer; Y : Integer ) : Real;
15269:   Function Real2Str( N : Real; Width, Places : integer ) : String;
15270:   Function mxStr2Comp( MyString : string ) : Comp;
15271:   Function mxStr2Pennies( S : String ) : Comp;
15272:   Function Str2Real( MyString : string ) : Real;
15273:   Function XToTheY( x, y : Real ) : Real;
15274: end;
15275:
15276: //*****Cindy Functions!*****
15277: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15278: begin
15279:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15280:     + '_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15281:     + 'cmAlterText_Html_RelatedAttach, cmAlterText_Html_Attach_RelatedAttach, cmReadNotification )');
15282:   MessagePlainText, 'String 'text/plain';
15283:   CL.AddConstantN('MessagePlainText_Attach', 'String 'multipart/mixed');
15284:   MessageAlterText_Html, 'String 'multipart/alternative';
15285:   MessageHtml_Attach, 'String 'multipart/mixed';
15286:   MessageHtml_RelatedAttach, 'String 'multipart/related; type="text/html";
15287:   MessageAlterText_Html_Attach, 'String 'multipart/mixed';
15288:   MessageAlterText_Html_RelatedAttach, 'String')('multipart/related; type="multipart/alternative";
15289:   MessageAlterText_Html_Attach_RelatedAttach, 'String 'multipart/mixed';
15290:   MessageReadNotification, 'String').( 'multipart/report; report-type="disposition-notification";
15291:   Function ForceDecodeHeader( aHeader : String ) : String;
15292:   Function Base64_EncodeString( Value : String; const aEncoding : TEncoding ) : string;
15293:   Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding ) : string;
15294:   Function Base64_DecodeToBytes( Value : String ) : TBytes;
15295:   Function IdHttp_DownloadFile(aSrcUrlFile, aDestFile: String; const OnWorkEvent: TWorkEvent): Boolean;
15296:   Function Get_MD5( const aFileName : string ) : string;
15297:   Function Get_MD5FromString( const aString : string ) : string;
15298: end;
15299:
15300: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15301: begin
15302:   Function IsFolder( SRec : TSearchrec ) : Boolean;
15303:   Function isFolderReadOnly( Directory : String ) : Boolean;
15304:   Function DirectoryIsEmpty( Directory : String ) : Boolean;
15305:   Function DirectoryWithSubDir( Directory : String ) : Boolean;
15306:   Procedure GetSubDirs( FromDirectory : String; aList : TStrings);
15307:   Function DiskFreeBytes( Drv : Char ) : Int64;
15308:   Function DiskBytes( Drv : Char ) : Int64;
15309:   Function GetFileBytes( Filename : String ) : Int64;
15310:   Function GetFilesBytes( Directory, Filter : String ) : Int64;
15311:   SE_CREATE_TOKEN_NAME, 'String 'SeCreateTokenPrivilege';
15312:   SE_ASSIGNPRIMARYTOKEN_NAME, 'String 'SeAssignPrimaryTokenPrivilege';
15313:   SE_LOCK_MEMORY_NAME, 'String 'SeLockMemoryPrivilege';
15314:   SE_INCREASE_QUOTA_NAME, 'String 'SeIncreaseQuotaPrivilege';
15315:   SE_UNSOLICITED_INPUT_NAME, 'String 'SeUnsolicitedInputPrivilege';
15316:   SE_MACHINE_ACCOUNT_NAME, 'String 'SeMachineAccountPrivilege';
15317:   SE_TCB_NAME, 'String 'SeTcbPrivilege';
15318:   SE_SECURITY_NAME, 'String 'SeSecurityPrivilege';
15319:   SE_TAKE_OWNERSHIP_NAME, 'String 'SeTakeOwnershipPrivilege';
15320:   SE_LOAD_DRIVER_NAME, 'String 'SeLoadDriverPrivilege';
15321:   SE_SYSTEM_PROFILE_NAME, 'String 'SeSystemProfilePrivilege';
15322:   SE_SYSTEMTIME_NAME, 'String 'SeSystemtimePrivilege';
15323:   SE_PROF_SINGLE_PROCESS_NAME, 'String 'SeProfileSingleProcessPrivilege';
15324:   SE_INC_BASE_PRIORITY_NAME, 'String 'SeIncreaseBasePriorityPrivilege';
15325:   SE_CREATE_PAGEFILE_NAME, 'String 'SeCreatePagefilePrivilege';
15326:   SE_CREATE_PERMANENT_NAME, 'String 'SeCreatePermanentPrivilege';
15327:   SE_BACKUP_NAME, 'String 'SeBackupPrivilege';
15328:   SE_RESTORE_NAME, 'String 'SeRestorePrivilege';
15329:   SE_SHUTDOWN_NAME, 'String 'SeShutdownPrivilege';
15330:   SE_DEBUG_NAME, 'String 'SeDebugPrivilege';
15331:   SE_AUDIT_NAME, 'String 'SeAuditPrivilege';
15332:   SE_SYSTEM_ENVIRONMENT_NAME, 'String 'SeSystemEnvironmentPrivilege';
15333:   SE_CHANGE_NOTIFY_NAME, 'String 'SeChangeNotifyPrivilege';
15334:   SE_REMOTE_SHUTDOWN_NAME, 'String 'SeRemoteShutdownPrivilege';
15335:   SE_UNDOCK_NAME, 'String 'SeUndockPrivilege';

```

```

15336: SE_SYNC_AGENT_NAME', 'String 'SeSyncAgentPrivilege');
15337: SE_ENABLE_DELEGATION_NAME', 'String 'SeEnableDelegationPrivilege');
15338: SE_MANAGE_VOLUME_NAME', 'String 'SeManageVolumePrivilege');
15339: end;
15340:
15341:
15342: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15343: begin
15344:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15345:     + 'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper '));
15346:   Function ShellGetExtensionName( FileName : String) : String);
15347:   Function ShellGetIconIndex( FileName : String) : Integer);
15348:   Function ShellGetIconHandle( FileName : String) : HIcon);
15349:   Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string));
15350:   Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string));
15351:   Procedure ShellRenameDir( DirFrom, DirTo : string);
15352:   Function cyShellExecute(Operation, FileName, Parameters, Directory: String; ShowCmd: Integer): Cardinal;
15353:   Procedure cyShellExecute1(ExeFilename, Parameters, ApplicationName, ApplicationClass: String; Restore: Boolean);
15354:   Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String);
15355:   Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string);
15356:   Function ShellExecuteEx(aFileName: string; const Parameters: string; const Directory: string; const
     WaitCloseCompletion : boolean) : Boolean);
15357:   Procedure RestoreAndSetForegroundWindow( Hnd : Integer);
15358:   Function RemoveDuplicatedPathDelimiter( Str : String) : String);
15359:   Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime);
15360:   Function GetModificationDate( Filename : String) : TDateTime);
15361:   Function GetCreationDate( Filename : String) : TDateTime);
15362:   Function GetLastAccessDate( Filename : String) : TDateTime);
15363:   Function FileDelete( Filename : String) : Boolean);
15364:   Function FileIsOpen( Filename : string) : boolean);
15365:   Procedure FilesDelete( FromDirectory : String; Filter : ShortString);
15366:   Function DirectoryDelete( Directory : String) : Boolean);
15367:   Function GetPrinters( PrintersList : TStrings) : Integer);
15368:   Procedure SetDefaultPrinter( PrinterName : String);
15369:   Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer);
15370:   Function WinToDosPath( WinPathName : String) : String);
15371:   Function DosToWinPath( DosPathName : String) : String);
15372:   Function cyGetWindowsVersion : TWindowsVersion);
15373:   Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean);
15374:   Procedure WindowsShutDown( Restart : boolean);
15375:   Procedure CreateShortCut( FileSrc, Parametres, FileLnk, Descript, DossierDeTravail,
     FileIcon: string; NumIcône: integer);
15376:   Procedure GetWindowsFonts( FontsList : TStrings);
15377:   Function GetAvailableFilename( DesiredFileName : String) : String);
15378: end;
15379:
15380: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15381: begin
15382:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey '));
15383:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15384:   Type(TStringRead', '( srFromLeft, srFromRight '));
15385:   Type(TStringReads', 'set of TStringRead');
15386:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive '));
15387:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar '));
15388:   Type(TWordsOptions', 'set of TWordsOption');
15389:   Type(TCarType', '( ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther '));
15390:   Type(TCarTypes', 'set of TCarType');
15391:   //CL.AddTypes('TUnicodeCategories', 'set of TUnicodeCategory');
15392:   CarTypeAlphabetic', 'LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15393:   Function Char_GetType( aChar : Char) : TCarType);
15394:   Function SubString_Count( Str : String; Separator : Char) : Integer);
15395:   Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer);
15396:   Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String);
15397:   Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer);
15398:   Procedure SubString_Add( var Str : String; Separator : Char; Value : String);
15399:   Procedure SubString_Insert( var Str: String; Separator: Char; SubStringIndex: Word; Value : String);
15400:   Procedure SubString_Edit( var Str: String; Separator: Char; SubStringIndex: Word; NewValue : String);
15401:   Function SubString_Remove( var Str: string; Separator: Char; SubStringIndex : Word) : Boolean);
15402:   Function SubString_Locate( Str: string; Separator: Char; SubString: String; Options: TStrLocateOptions): Integer;
15403:   Function SubString_Ribbon( Str : string; Separator : Char; Current : Word; MoveBy : Integer);
15404:   Function SubString_Ribbon1( Str: string; Separator: Char; Current: String; MoveBy: Integer): String);
15405:   Function String_Quote( Str : String) : String);
15406:   Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char);
15407:   Function String_ExtractCars( fromStr: String; CarTypes: TCarTypes; IncludeCars, ExcludeCars: String): String;
15408:   Function String_GetWord( Str : String; StringRead : TStringRead) : String);
15409:   Function String_GetInteger( Str : String; StringRead : TStringRead) : String);
15410:   Function String_ToInt( Str : String) : Integer);
15411:   Function String_Uppercase( Str : String; Options : TWordsOptions) : String);
15412:   Function String_Lowercase( Str : String; Options : TWordsOptions) : String);
15413:   Function String_Reverse( Str : String) : String);
15414:   Function String_Pos( SubStr: String; Str: String; fromPos: Integer; CaseSensitive: TCaseSensitive) Integer;
15415:   Function String_Pos1( SubStr: String; Str: String; StringRead: TStringRead; Occurrence: Word; CaseSensitive :
     TCaseSensitive): Integer);
15416:   Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String);
15417:   Function String_Copy1( Str: String; StringRead: TStringRead; UntilFind: String; _Inclusive: Boolean): String;
15418:   Function String_Copy2( Str: String; Between1: String; Between1MustExist: Boolean; Between2: String;
     Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String);
15419:   Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String);
15420:   Function String_Delete1( Str : String; delStr : String; CaseSensitive: TCaseSensitive) : String);

```

```

15421: Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String'';
15422: Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String'';
15423: Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String'';
15424: Function String_End( Str : String; Cars : Word) : String'';
15425: Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
AlwaysFindFromBeginning : Boolean) : String'';
15426: Function String_SubstCar( Str : String; Old, New : Char) : String'';
15427: Function String_Count( Str : String; SubStr : String; CaseSensitive : TCaseSensitive) : Integer'';
15428: Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15429: Function String_IsNumbers( Str : String) : Boolean'';
15430: Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer'';
15431: Function StringToCsvCell( aStr : String) : String'';
15432: end;
15433:
15434: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15435: begin
15436:   Function LongDayName( aDate : TDate) : String'';
15437:   Function LongMonthName( aDate : TDate) : String'';
15438:   Function ShortYearOf( aDate : TDate) : byte'';
15439:   Function DateToStrYYYYMMDD( aDate : TDate) : String'';
15440:   Function StrYYYYMMDDToDate( aStr : String) : TDate'';
15441:   Function SecondsToMinutes( Seconds : Integer) : Double'';
15442:   Function MinutesToSeconds( Minutes : Double) : Integer'';
15443:   Function MinutesToHours( Minutes : Integer) : Double'';
15444:   Function HoursToMinutes( Hours : Double) : Integer'';
15445:   Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime'';
15446:   Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)'';
15447:   Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime'';
15448:   Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64'';
15449:   Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15450:   Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64'';
15451:   Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var RsltBegin :
TDateTime; RsltEnd : TDateTime) : Boolean'';
15452:   Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15453:   Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean'';
15454: end;
15455:
15456: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15457: begin
15458:   Type(TStringsSortType,'(' stNone, stStringSensitive, stStringInsensitive, stExtended)'';
15459:   Type(TStringsValueKind,'(' skStringSensitive, skStringInsensitive, skExtended)'';
15460:   Type(TcyLocateOption,'(' lCaseInsensitive, lPartialKey)'';
15461:   Type(TcyLocateOptions,'set of TcyLocateOption'');
15462:   Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer'';
15463:   Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer'';
15464:   Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer'';
15465:   Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)'';
15466:   Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)'';
15467:   Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode'';
15468:   Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode'';
15469:   Function
TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode'';
15470:   Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode'';
15471:   Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
CopySubChildren:Bool;
15472:   Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)'';
15473:   Procedure RichEditStringReplace( aRichEdit : TRichEdit; OldPattern, NewPattern : string; Flags :
TReplaceFlags)'';
15474:   Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl'';
15475:   Procedure cyCenterControl( aControl : TControl)'';
15476:   Function GetLastParent( aControl : TControl) : TWinControl'';
15477:   Function GetControlBitmap( aControl : TWinControl) : TBitmap'';
15478:   Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap'';
15479: end;
15480:
15481: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15482: begin
15483:   Function TablePackTable( Tab : TTable) : Boolean'';
15484:   Function TableRegenIndexes( Tab : TTable) : Boolean'';
15485:   Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean'';
15486:   Function TableUndeleteRecord( Tab : TTable) : Boolean'';
15487:   Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15488:   Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean'';
15489:   Function TableEmptyTable( Tab : TTable) : Boolean'';
15490:   Function TableFindKey( aTable : TTable; Value : String) : Boolean'';
15491:   Procedure TableFindNearest( aTable : TTable; Value : String)'';
15492:   Function
TableCreate(Owner:TComponent;DataBaseName:ShortString;TableName:String;IndexName:ShortString;ReadOnly :
Boolean):TTable;
15493:   Function
TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15494:   Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String'';
15495: end;
15496:
15497: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15498: begin
15499:   SIRegister_TcyRunTimeDesign(CL);
15500:   SIRegister_TcyShadowText(CL);
15501:   SIRegister_TcyBgPicture(CL);

```



```

15502:   SIRegister_TcyGradient(CL);
15503:   SIRegister_tcyBevel(CL);
15504:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15505:   SIRegister_tcyBevels(CL);
15506:   SIRegister_TcyImageListOptions(CL);
15507:   Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture);
15508: end;
15509:
15510: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15511: begin
15512:   Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
Maxdegrade : Byte;
15513:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap);
15514:   Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrade : byte);
15515:   Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrade:byte);
15516:   Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrade :
Byte; toRect : TRect; OrientationShape : TDgradOrientationShape);
15517:   Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrade :
Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap);
15518:   Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer);
15519:   Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);
15520:   Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
BottomColor : TColor; const Width : Integer; const RoundRect : boolean);
15521:   Procedure cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,
BottomRightColor:TColor;Width:Integer;const DrawLeft:Boolean;const DrawTop:Bool;const DrawRight:Bool;const
DrawBottom:Bool;const RoundRect:bool;
15522:   Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
aState : TButtonState; Focused, Hot : Boolean);
15523:   Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
aState : TButtonState; Focused, Hot : Boolean);
15524:   Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
TColor; aState : TButtonState; Focused, Hot : Boolean);
15525:   Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean);
15526:   Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor);
15527:   Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer);
15528:   Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15529:   Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;
15530:   Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt);
15531:   Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;
15532:   Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;
15533:   Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout);
15534:   Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord;
15535:   Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord;
15536:   Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean);
15537:   Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean);
15538:   Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean);
15539:   Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean);
15540:   Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);
15541:   Procedure DrawCanvas1(Destination:TCanvas; DestRect:TRect;Src:TCanvas; SrcRect : TRect; TransparentColor
: TColor; const aStyle:TBgStyle; const aPosition:TBgPosition; const IndentX : Integer; const IndentY :
Integer;
15542:   const IntervalX : Integer; const IntervalY : Integer; const RepeatX : Integer; const RepeatY :
Integer);
15543:   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;
15544:   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);
15545:   Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap);
15546:   Function ValidGraphic( aGraphic : TGraphic) : Boolean);
15547:   Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor);
15548:   Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor);
15549:   Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor);
15550:   Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor);
15551:   Function MediumColor( Color1, Color2 : TColor) : TColor);
15552:   Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect);
15553:   Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect);
15554:   Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect);
15555:   Procedure InflateRectPercent( var aRect : TRect; withPercent : Double);
15556:   Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect);
15557:   Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect);
15558:   Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean);
15559:   Function PointInEllipse( const aPt : TPoint; const aRect : TRect) : boolean);
15560:   Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer);
15561: end;
15562:
15563: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);

```

```

15564: begin
15565:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15566:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15567:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15568:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15569:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15570:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight, '
15571:     + ' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft )');
15572:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15573:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15574:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15575:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15576:   Type(TDgradBalanceMode(bmNormal, bmMirror, bmReverse, bmReverseFromColor, bmInvertReverse,
bmInvertReverseFromColor);
15577:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
rjResizeTopLeft, rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15578:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15579:   cCaptionOrientationWarning', 'String')('Note that text orientation doesn't work with all fonts!');
15580: end;
15581:
15582: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15583: begin
15584:   Const SERVICES_ACTIVE_DATABASEA', 'String 'ServicesActive');
15585:   SERVICES_ACTIVE_DATABASEW', 'String') 'ServicesActive');
15586:   Const SERVICES_ACTIVE_DATABASE', 'String') 'SERVICES_ACTIVE_DATABASE');
15587:   Const SERVICES_FAILED_DATABASEA', 'String' 'ServicesFailed');
15588:   Const SERVICES_FAILED_DATABASEW', 'String' 'ServicesFailed');
15589:   Const SERVICES_FAILED_DATABASE', 'String' 'SERVICES_FAILED_DATABASE');
15590:   Const SC_GROUP_IDENTIFIER', 'String') ('+');
15591:   Const SC_GROUP_IDENTIFIERW', 'String') ('+');
15592:   Const SC_GROUP_IDENTIFIER', 'string 'SC_GROUP_IDENTIFIER');
15593:   Const SERVICE_NO_CHANGE', 'LongWord $FFFFFFF);
15594:   Const SERVICE_ACTIVE', 'LongWord') ($00000001);
15595:   Const SERVICE_INACTIVE', 'LongWord $00000002);
15596:   Const SERVICE_CONTROL_STOP', 'LongWord $00000001);
15597:   Const SERVICE_CONTROL_PAUSE', 'LongWord $00000002);
15598:   Const SERVICE_CONTROL_CONTINUE', 'LongWord $00000003);
15599:   Const SERVICE_CONTROL_INTERROGATE', 'LongWord $00000004);
15600:   Const SERVICE_CONTROL_SHUTDOWN', 'LongWord $00000005);
15601:   Const SERVICE_STOPPED', 'LongWord $00000001);
15602:   Const SERVICE_START_PENDING', 'LongWord $00000002);
15603:   Const SERVICE_STOP_PENDING', 'LongWord $00000003);
15604:   Const SERVICE_RUNNING', 'LongWord $00000004);
15605:   Const SERVICE_CONTINUE_PENDING', 'LongWord $00000005);
15606:   Const SERVICE_PAUSE_PENDING', 'LongWord $00000006);
15607:   Const SERVICE_PAUSED', 'LongWord $00000007);
15608:   Const SERVICE_ACCEPT_STOP', 'LongWord $00000001);
15609:   Const SERVICE_ACCEPT_PAUSE_CONTINUE', 'LongWord $00000002);
15610:   Const SERVICE_ACCEPT_SHUTDOWN', 'LongWord $00000004);
15611:   Const SC_MANAGER_CONNECT', 'LongWord $0001);
15612:   Const SC_MANAGER_CREATE_SERVICE', 'LongWord $0002);
15613:   Const SC_MANAGER_ENUMERATE_SERVICE', 'LongWord $0004);
15614:   Const SC_MANAGER_LOCK', 'LongWord $0008);
15615:   Const SC_MANAGER_QUERY_LOCK_STATUS', 'LongWord $0010);
15616:   Const SC_MANAGER_MODIFY_BOOT_CONFIG', 'LongWord $0020);
15617:   Const SERVICE_QUERY_CONFIG', 'LongWord $0001);
15618:   Const SERVICE_CHANGE_CONFIG', 'LongWord $0002);
15619:   Const SERVICE_QUERY_STATUS', 'LongWord $0004);
15620:   Const SERVICE_ENUMERATE_DEPENDENTS', 'LongWord $0008);
15621:   Const SERVICE_START', 'LongWord $0010);
15622:   Const SERVICE_STOP', 'LongWord $0020);
15623:   Const SERVICE_PAUSE_CONTINUE', 'LongWord $0040);
15624:   Const SERVICE_INTERROGATE', 'LongWord $0080);
15625:   Const SERVICE_USER_DEFINED_CONTROL', 'LongWord $0100);
15626:   Const SERVICE_KERNEL_DRIVER', 'LongWord $00000001);
15627:   Const SERVICE_FILE_SYSTEM_DRIVER', 'LongWord $00000002);
15628:   Const SERVICE_ADAPTER', 'LongWord $00000004);
15629:   Const SERVICE_RECOGNIZER_DRIVER', 'LongWord $00000008);
15630:   Const SERVICE_WIN32_OWN_PROCESS', 'LongWord $00000010);
15631:   Const SERVICE_WIN32_SHARE_PROCESS', 'LongWord $00000020);
15632:   Const SERVICE_INTERACTIVE_PROCESS', 'LongWord $00000100);
15633:   Const SERVICE_BOOT_START', 'LongWord $00000000);
15634:   Const SERVICE_SYSTEM_START', 'LongWord $00000001);
15635:   Const SERVICE_AUTO_START', 'LongWord $00000002);
15636:   Const SERVICE_DEMAND_START', 'LongWord $00000003);
15637:   Const SERVICE_DISABLED', 'LongWord $00000004);
15638:   Const SERVICE_ERROR_IGNORE', 'LongWord $00000000);
15639:   Const SERVICE_ERROR_NORMAL', 'LongWord $00000001);
15640:   Const SERVICE_ERROR_SEVERE', 'LongWord $00000002);
15641:   Const SERVICE_ERROR_CRITICAL', 'LongWord $00000003);
15642:   CL.AddTypeS('SC_HANDLE', 'THandle');
15643:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15644:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15645:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15646:     + ': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe
15647:     + 'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15648:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15649:   Const TServiceStatus', '_SERVICE_STATUS');
15650:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis

```

```

15651:   +'playName : PChar; ServiceStatus : TServiceStatus; end');
15652:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15653:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15654:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15655:   TEnumServiceStatus', 'TEnumServiceStatusA');
15656:   SC_LOCK', '___Pointer');
15657:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOwner: PChar;dwLockDuration:DWORD;end;
15658:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15659:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15660:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15661:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15662:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15663:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15664:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType: DWORD; dwStartT'
15665:   +'type: DWORD; dwErrorControl: DWORD; lpBinaryPathName: PChar; lpLoadO'
15666:   +'rderGroup: PChar; dwTagId: DWORD; lpDependencies: PChar; lpServ'
15667:   +'iceStartName: PChar; lpDisplayName: PChar; end');
15668:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15669:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15670:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15671:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15672:   TQueryServiceConfig', 'TQueryServiceConfigA');
15673:   Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL;
15674:   Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15675:   Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15676:   +' lpdwTagId: DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15677:   Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15678:   +'lpdwTagId: DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15679:   Function DeleteService( hService : SC_HANDLE) : BOOL;
15680:   Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL;
15681:   Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
: DWORD):BOOL;
15682:   Function GetServiceKeyName(hSCManager:SC_HANDLE;lpDisplayName,lpServiceName:PChar;var
lpchBuffer:DWORD):BOOL;
15683:   Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceName,lpDisplayName:PChar;var
lpchBuffer:DWORD):BOOL;
15684:   Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK;
15685:   Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL;
15686:   Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE;
15687:   Function OpenService( hSCManager : SC_HANDLE; lpServiceName : PChar; dwDesiredAccess : DWORD) :
SC_HANDLE;
15688:   Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL;
15689:   Function QueryServiceStatus( hService : SC_HANDLE; var lpServiceStatus : TServiceStatus) : BOOL;
15690:   Function SetServiceStatus( hServiceStatus : SERVICE_STATUS_HANDLE; var lpServiceStatus : TServiceStatus)
: BOOL;
15691:   Function StartService( hService : SC_HANDLE; dwNumServiceArgs : DWORD; var lpServiceArgVectors : PChar) :
BOOL;
15692:   Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL;
15693: end;
15694:
15695: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15696: begin
15697:   CL.AddDelphiFunction('Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption :
TCaption; AStartOfWeek : TDayOfWeekName; AWeekends : TDaysOfWeek; AWeekendColor : TColor; BtnHints :
TStrings; MinDate : TDateTime; MaxDate : TDateTime) : Boolean');
15698:   Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName; AWeekends:TDaysOfWeek;
AWeekendColor:TColor;BtnHints:TStrings;MinDate:TDateTime;MaxDate:TDateTime):Boolean;
15699:   Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15700:   Function CreatePopupCalendar(AOwner:TComponent;ABidiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
TWinControl;
15701:   Procedure SetupPopupCalendar( PopupCalendar: TWinControl; AStartOfWeek : TDayOfWeekName; AWeekends :
TDaysOfWeek; AWeekendColor : TColor; BtnHints : TStrings; FourDigitYear : Boolean; MinDate : TDateTime;
MaxDate : TDateTime);
15702:   Function CreateNotifyThread( const FolderName : string; WatchSubtree : Boolean; Filter :
TFileChangeFilters) : TJvNotifyThread;
15703: end;
15704:
15705: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15706: begin
15707:   CL.AddClassN(CL.FindClass('TObject'),'EJclNtfsError');
15708:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15709:   Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;
15710:   Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;
15711:   Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean;
15712:   Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState);
15713:   Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState);
15714:   Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState);
15715:   Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool);
15716:   Function NtfsSetSparse2( const FileName : string) : Boolean;
15717:   Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean;
15718:   Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean;
15719:   Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean;
15720:   Function NtfsGetSparse2( const FileName : string) : Boolean;

```

```

15721: Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean'';
15722: Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean'';
15723: Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean'';
15724: Function NtfsReparsePointsSupported2( const Volume : string) : Boolean'';
15725: Function NtfsFileHasReparsePoint2( const Path : string) : Boolean'';
15726: Function NtfsIsFolderMountPoint2( const Path : string) : Boolean'';
15727: Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean'';
15728: Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean'';
15729: CL.AddTypeS('TOpenLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15730: Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean'';
15731: Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean'';
15732: Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean'';
15733: Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean'';
15734: Function NtfsRequestOpLock2( Handle : THandle; Kind : TopLock; Overlapped : TOverlapped) : Boolean'';
15735: Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean'';
15736: Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean'';
15737: Function NtfsGetJunctionPointDestination2( const Source : string; var Destination : string) : Boolean'';
15738: CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15739:   + 'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )');
15740: CL.AddTypeS('TStreamIds', 'set of TStreamId');
15741: TInternalFindStreamData, 'record FileHandle:THandle; Context: TObject; StreamIds : TStreamIds; end');
15742: CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15743:   + 'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15744: Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15745: Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean'';
15746: Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean'';
15747: Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean'';
15748: Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean'';
15749: Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean'';
15750: CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15751: Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean'';
15752: Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
List:TStrings):Bool
15753: Function NtfsDeleteHardLinks2( const FileName : string) : Boolean'';
15754: FindClass('TOBJECT'), 'EJclFileSummaryError');
15755: TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15756: TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15757: TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean'';
15758: CL.AddClassN(CL.FindClass('TOBJECT'), 'TJclFileSummary');
15759: SIRegister_TJclFilePropertySet(CL);
15760: //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15761: SIRegister_TJclFileSummary(CL);
15762: SIRegister_TJclFileSummaryInformation(CL);
15763: SIRegister_TJclDocSummaryInformation(CL);
15764: SIRegister_TJclMediaFileSummaryInformation(CL);
15765: SIRegister_TJclMSISummaryInformation(CL);
15766: SIRegister_TJclShellSummaryInformation(CL);
15767: SIRegister_TJclStorageSummaryInformation(CL);
15768: SIRegister_TJclImageSummaryInformation(CL);
15769: SIRegister_TJclDisplacedSummaryInformation(CL);
15770: SIRegister_TJclBriefCaseSummaryInformation(CL);
15771: SIRegister_TJclMiscSummaryInformation(CL);
15772: SIRegister_TJclWebViewSummaryInformation(CL);
15773: SIRegister_TJclMusicSummaryInformation(CL);
15774: SIRegister_TJclDRMSummaryInformation(CL);
15775: SIRegister_TJclVideoSummaryInformation(CL);
15776: SIRegister_TJclAudioSummaryInformation(CL);
15777: SIRegister_TJclControlPanelSummaryInformation(CL);
15778: SIRegister_TJclVolumeSummaryInformation(CL);
15779: SIRegister_TJclShareSummaryInformation(CL);
15780: SIRegister_TJclLinkSummaryInformation(CL);
15781: SIRegister_TJclQuerySummaryInformation(CL);
15782: SIRegister_TJclImageInformation(CL);
15783: SIRegister_TJclJpegSummaryInformation(CL);
15784: end;
15785:
15786: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15787: begin
15788:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15789:   T8087Rounding', '( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15790:   T8087Infinity', '( icProjective, icAffine )');
15791:   T8087Exception', '( emInvalidOp, emDenormalizedOperand, emZeroDivide, emOverflow, emUnderflow, emPrecision';
15792:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15793: Function Get8087ControlWord : Word'';
15794: Function Get8087Infinity : T8087Infinity'';
15795: Function Get8087Precision : T8087Precision'';
15796: Function Get8087Rounding : T8087Rounding'';
15797: Function Get8087StatusWord( ClearExceptions : Boolean) : Word'';
15798: Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity'';
15799: Function Set8087Precision( const Precision : T8087Precision) : T8087Precision'';
15800: Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding'';
15801: Function Set8087ControlWord( const Control : Word) : Word'';
15802: Function ClearPending8087Exceptions : T8087Exceptions'';
15803: Function GetPending8087Exceptions : T8087Exceptions'';
15804: Function GetMasked8087Exceptions : T8087Exceptions'';
15805: Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions'';
15806: Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions'';
15807: Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions'';
15808: end;

```



```

15809:
15810:
15811:
15812: Functions_max hex in the box maXbox
15813: functionslist.txt
15814: FunctionsList1 3.9.9.86/88/91/92/94
15815:
15816: *****
15817: Procedure
15818: PROCEDURE SIZE 7507 7401 6792 6310 5971 4438 3797 3600
15819: Procedure *****Now the Procedure list*****
15820: Procedure ( ACol, ARow : Integer; Items : TStrings)
15821: Procedure ( Agg : TAggregate)
15822: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
15823: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
15824: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
15825: Procedure ( ASender : TObjekt; const ABytes : Integer)
15826: Procedure ( ASender : TObjekt; VStream : TStream)
15827: Procedure ( AThread : TIdThread)
15828: Procedure ( AWebModule : TComponent)
15829: Procedure ( Column : TColumn)
15830: Procedure ( const AUsername : String; const APassord : String; AAuthenticationResult : Boolean)
15831: Procedure ( const iStart : integer; const sText : string)
15832: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
15833: Procedure ( Database : TDatabase; LoginParams : TStrings)
15834: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:
TReconcileAction)
15835: Procedure ( DATASET : TDATASET)
15836: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
15837: Procedure ( DATASET : TDATASET; E : TObjekt; var ACTION : TDATAACTION)
15838: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
15839: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
15840: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
15841: Procedure ( Done : Integer)
15842: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
15843: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
15844: Procedure
(HeaderControl:TCustomHeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
15845: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
15846: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
15847: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
15848: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
15849: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
15850: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
15851: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
15852: Procedure ( SENDER : TFIELD; const TEXT : String)
15853: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
15854: Procedure ( Sender : TIdTelnet; const Buffer : String)
15855: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
15856: Procedure ( SENDER : TObjekt; ACANVAS : TCanvas; ARECT : TRect; SELECTED : BOOLEAN)
15857: Procedure ( SENDER : TObjekt; ACANVAS : TCanvas; ARECT : TRect; STATE : TOWNERDRAWSTATE)
15858: Procedure ( SENDER : TObjekt; ACANVAS : TCanvas; var WIDTH, HEIGHT : INTEGER)
15859: Procedure ( Sender : TObjekt; ACol, ARow : Longint; const Value : string)
15860: Procedure ( Sender : TObjekt; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
15861: Procedure ( Sender : TObjekt; ACol, ARow : Longint; var CanSelect : Boolean)
15862: Procedure ( Sender : TObjekt; ACol, ARow : Longint; var Value : string)
15863: Procedure ( Sender : TObjekt; Button : TMPBtnType)
15864: Procedure ( Sender : TObjekt; Button : TMPBtnType; var DoDefault : Boolean)
15865: Procedure ( Sender : TObjekt; Button : TUDBtnType)
15866: Procedure ( Sender : TObjekt; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
15867: Procedure ( Sender : TObjekt; ClientSocket : TServerClientWinSocket; var SocketThread : TServerClientThread)
15868: Procedure ( Sender : TObjekt; Column : TListColumn)
15869: Procedure ( Sender : TObjekt; Column : TListColumn; Point : TPoint)
15870: Procedure ( Sender : TObjekt; Connecting : Boolean)
15871: Procedure (Sender:TObjekt;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
DoneDraw:Bool
15872: Procedure (Sender:TObjekt; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
15873: Procedure ( Sender : TObjekt; const Rect : TRect; Field : TField; State : TGridDrawState)
15874: Procedure (Sender: TObjekt; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
15875: Procedure ( Sender : TObjekt; E : Exception; var Handled : Boolean)
15876: Procedure ( Sender : TObjekt; FromIndex, ToIndex : Longint)
15877: Procedure ( Sender : TObjekt; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
15878: Procedure ( Sender : TObjekt; Index : Longint)
15879: Procedure ( Sender : TObjekt; Item : TListItem)
15880: Procedure ( Sender : TObjekt; Item : TListItem; Change : TItemChange)
15881: Procedure ( Sender : TObjekt; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
15882: Procedure ( Sender : TObjekt; Item : TListItem; Selected : Boolean)
15883: Procedure ( Sender : TObjekt; Item : TListItem; var AllowEdit : Boolean)
15884: Procedure ( Sender : TObjekt; Item : TListItem; var S : string)
15885: Procedure ( Sender : TObjekt; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
15886: Procedure ( Sender : TObjekt; ModalResult : TModalResult; var CanClose : Boolean)
15887: Procedure ( Sender : TObjekt; Month : LongWord; var MonthBoldInfo : LongWord)
15888: Procedure ( Sender : TObjekt; NewTab : Integer; var AllowChange : Boolean)
15889: Procedure ( Sender : TObjekt; Node : TTreeNode)
15890: Procedure ( Sender : TObjekt; Node : TTreeNode; var AllowChange : Boolean)
15891: Procedure ( Sender : TObjekt; Node : TTreeNode; var AllowCollapse : Boolean)
15892: Procedure ( Sender : TObjekt; Node : TTreeNode; var AllowEdit : Boolean)
15893: Procedure ( Sender : TObjekt; Node : TTreeNode; var AllowExpansion : Boolean)
15894: Procedure ( Sender : TObjekt; Node : TTreeNode; var S : string)

```

```

15895: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
15896: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
15897: Procedure ( Sender : TObject; Rect : TRect)
15898: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
15899: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
15900: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
15901: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
15902: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
15903: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
15904: Procedure ( SENDER : TObject; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
15905: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
15906: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
15907: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
15908: Procedure ( Sender : TObject; Thread : TServerClientThread)
15909: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
15910: Procedure ( Sender : TObject; Username, Password : string)
15911: Procedure ( Sender : TObject; var AllowChange : Boolean)
15912: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
15913: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
15914: Procedure ( Sender : TObject; var Continue : Boolean)
15915: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TidHTTPMethod)
15916: Procedure ( Sender : TObject; var Username : string)
15917: Procedure ( Sender : TObject; Wnd : HWND)
15918: Procedure ( Sender : TToolBar; Button : TToolButton)
15919: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
15920: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
15921: Procedure ( Sender : TToolBar; Index : Integer; var Allow : Boolean)
15922: Procedure ( Sender : TToolBar; Index : Integer; var Button : TToolButton)
15923: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
15924: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
15925: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
15926: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
15927: procedure (Sender: TObject)
15928: procedure (Sender: TObject; var Done: Boolean)
15929: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
15930: procedure _T(Name: tbtString; v: Variant);
15931: Procedure AbandonSignalHandler( RtlSigNum : Integer)
15932: Procedure Abort
15933: Procedure About1Click( Sender : TObject)
15934: Procedure Accept( Socket : TSocket)
15935: Procedure AESSymmetricExecute(const plaintext, ciphertext, password: string)
15936: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
15937: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
15938: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
15939: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
15940: Procedure Add( Addend1, Addend2 : TMyBigInt)
15941: Procedure ADD( const AKEY, AVALUE : VARIANT)
15942: Procedure Add( const Key : string; Value : Integer)
15943: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXTIONS)
15944: Procedure ADD( FIELD : TFIELD)
15945: Procedure ADD( ITEM : TMENUITEM)
15946: Procedure ADD( POPUP : TPOPUPMENU)
15947: Procedure AddCharacters( xCharacters : TCharSet)
15948: Procedure AddDriver( const Name : string; List : TStringList)
15949: Procedure AddImages( Value : TCustomImageList)
15950: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
15951: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
15952: Procedure AddLoader( Loader : TBitmapLoader)
15953: Procedure ADDPARAM( VALUE : TPARAM)
15954: Procedure AddPassword( const Password : string)
15955: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
15956: Procedure AddState( oState : TniRegularExpressionState)
15957: Procedure AddStrings( Strings : TStringList);
15958: procedure AddStrings(Strings: TStringList);
15959: Procedure AddStrings1( Strings : TWideStrings);
15960: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
15961: Procedure AddToRecentDocs( const Filename : string)
15962: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharSet)
15963: Procedure AllFunctionsList1Click( Sender : TObject)
15964: procedure AllObjectsList1Click(Sender: TObject);
15965: Procedure Allocate( AAllocateBytes : Integer)
15966: procedure AllResourceList1Click(Sender: TObject);
15967: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
15968: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
15969: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
15970: Procedure AnsiFree( var s : AnsiString)
15971: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
15972: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
15973: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
15974: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
15975: Procedure AntiFreeze;
15976: Procedure APPEND
15977: Procedure Append( const S : WideString)
15978: procedure Append(S: string);
15979: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
15980: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
15981: Procedure AppendChunk( Val : OleVariant)
15982: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
15983: Procedure AppendStr( var Dest : string; S : string)

```

```

15984: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
15985: Procedure ApplyRange
15986: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
15987: Procedure Arrange( Code : TListArrangement)
15988: procedure Assert(expr : Boolean; const msg: string);
15989: procedure Assert2(expr : Boolean; const msg: string);
15990: Procedure Assign( AList : TCustomBucketList)
15991: Procedure Assign( Other : TObject)
15992: Procedure Assign( Source : TDragObject)
15993: Procedure Assign( Source : TPersistent)
15994: Procedure Assign(Source: TPersistent)
15995: procedure Assign2(mystring, mypath: string);
15996: Procedure AssignCurValues( Source : TDataSet);
15997: Procedure AssignCurValues1( const CurValues : Variant);
15998: Procedure ASSIGNFIELD( FIELD : TFIELD)
15999: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
16000: Procedure AssignFile(var F: Text; FileName: string)
16001: procedure AssignFile(var F: TextFile; FileName: string)
16002: procedure AssignFileRead(var mystring, myfilename: string);
16003: procedure AssignFileWrite(mystring, myfilename: string);
16004: Procedure AssignTo( Other : TObject)
16005: Procedure AssignValues( Value : TParameters)
16006: Procedure ASSIGNVALUES( VALUE : TPARAMS)
16007: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
16008: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
16009: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
16010: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
16011: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
16012: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
16013: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
16014: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
16015: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
16016: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
16017: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
16018: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
16019: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
16020: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
16021: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
16022: procedure Beep
16023: Procedure BeepOk
16024: Procedure BeepQuestion
16025: Procedure BeepHand
16026: Procedure BeepExclamation
16027: Procedure BeepAsterisk
16028: Procedure BeepInformation
16029: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
16030: Procedure BeginLayout
16031: Procedure BeginTimer( const Delay, Resolution : Cardinal)
16032: Procedure BeginUpdate
16033: procedure BeginUpdate;
16034: procedure BigScreen1Click(Sender: TObject);
16035: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
16036: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
16037: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
16038: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
16039: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
16040: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
16041: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
16042: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
16043: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
16044: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
16045: Procedure BreakPointMenuClick( Sender : TObject)
16046: procedure BRINGTOFRONT
16047: procedure BringToFront;
16048: Procedure btnBackClick( Sender : TObject)
16049: Procedure btnBrowseClick( Sender : TObject)
16050: Procedure BtnClick( Index : TNavigateBtn)
16051: Procedure btnLargeIconsClick( Sender : TObject)
16052: Procedure BuildAndSendRequest( AURI : TIdURI)
16053: Procedure BuildCache
16054: Procedure BurnMemory( var Buff, BuffLen : integer)
16055: Procedure BurnMemoryStream( Destructo : TMemoryStream)
16056: Procedure CalculateFirstSet
16057: Procedure Cancel
16058: procedure CancelDrag;
16059: Procedure CancelEdit
16060: procedure CANCELHINT
16061: Procedure CancelRange
16062: Procedure CancelUpdates
16063: Procedure CancelWriteBuffer
16064: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
16065: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
16066: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
16067: procedure CaptureScreenFormat(vname: string; vextension: string);
16068: procedure CaptureScreenPNG(vname: string);
16069: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
16070: procedure CASCADE
16071: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
16072: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);

```

```

16073: Procedure cbPathClick( Sender : TObject)
16074: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
16075: Procedure cedebugAfterExecute( Sender : TPSScript)
16076: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
16077: Procedure cedebugCompile( Sender : TPSScript)
16078: Procedure cedebugExecute( Sender : TPSScript)
16079: Procedure cedebugIdle( Sender : TObject)
16080: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
16081: Procedure CenterHeight( const pc, pcParent : TControl)
16082: Procedure CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
16083: Procedure CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
16084: Procedure Change
16085: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
16086: Procedure Changed
16087: Procedure ChangeDir( const ADirName : string)
16088: Procedure ChangeDirUp
16089: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
16090: Procedure ChangeLevelBy( Value : TChangeRange)
16091: Procedure ChDir(const s: string)
16092: Procedure Check(Status: Integer)
16093: Procedure CheckCommonControl( CC : Integer)
16094: Procedure CHECKFIELDNAME( const FIELDNAME : String)
16095: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
16096: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
16097: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
16098: Procedure CheckToken( T : Char)
16099: procedure CheckToken(t:char)
16100: Procedure CheckTokenSymbol( const S : string)
16101: procedure CheckTokenSymbol(s:string)
16102: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
16103: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
16104: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
16105: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
16106: procedure CipherFileClick(Sender: TObject);
16107: Procedure Clear;
16108: Procedure Clear1Click( Sender : TObject)
16109: Procedure ClearColor( Color : TColor)
16110: Procedure CLEARITEM( AITEM : TMENUITEM)
16111: Procedure ClearMapping
16112: Procedure ClearSelection( KeepPrimary : Boolean)
16113: Procedure ClearWriteBuffer
16114: Procedure Click
16115: Procedure Close
16116: Procedure Close1Click( Sender : TObject)
16117: Procedure CloseDatabase( Database : TDatabase)
16118: Procedure CloseDataSets
16119: Procedure CloseDialog
16120: Procedure CloseFile(var F: Text);
16121: Procedure Closure
16122: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16123: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16124: Procedure CodeCompletionList1Click( Sender : TObject)
16125: Procedure ColEnter
16126: Procedure Collapse
16127: Procedure Collapse( Recurse : Boolean)
16128: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
16129: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
16130: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
16131: Procedure Compile1Click( Sender : TObject)
16132: procedure ComponentCount1Click(Sender: TObject);
16133: Procedure Compress(azipfolder, azipfile: string)
16134: Procedure DeCompress(azipfolder, azipfile: string)
16135: Procedure XZip(azipfolder, azipfile: string)
16136: Procedure XUnZip(azipfolder, azipfile: string)
16137: Procedure Connect( const ATimeout : Integer)
16138: Procedure Connect( Socket : TSocket)
16139: procedure Console1Click(Sender: TObject);
16140: Procedure Continue
16141: Procedure ContinueCount( var Counter : TJclCounter)
16142: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
16143: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
16144: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
16145: Procedure ConvertImage(vsource, vdestination: string);
16146: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
16147: Procedure ConvertToGray(Cnv: TCanvas);
16148: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
16149: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
16150: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
16151: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
16152: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
16153: Procedure CopyFrom( mbCopy : TMyBigInt)
16154: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
16155: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
16156: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array of Byte; const ADestIndex : Integer; const ALength : Integer)
16157: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const ALen:Int)
16158: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
16159: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)

```



```

16160: Procedure CopyTidIPv6Address(const ASource:TidIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
16161: Procedure CopyTidLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
16162: Procedure CopyTidNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
16163: Procedure CopyTidNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
16164: Procedure CopyTidString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
16165: Procedure CopyTidWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
16166: Procedure CopyToClipboard
16167: Procedure CountParts
16168: Procedure CreateDataSet
16169: Procedure CreateEmptyFile( const FileName : string)
16170: Procedure CreateFileFromString( const FileName, Data : string)
16171: Procedure CreateFromDelta( Source : TPacketDataSet)
16172: procedure CREATEHANDLE
16173: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
16174: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
16175: Procedure CreateTable
16176: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
16177: procedure CSyntaxClick(Sender: TObject);
16178: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
16179: Procedure CURSORPOSCHANGED
16180: procedure CutFirstDirectory(var S: String)
16181: Procedure DataBaseError(const Message: string)
16182: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
16183: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
16184: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
16185: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
16186: Procedure DBIError(errorCode: Integer)
16187: Procedure DebugOutput( const AText : string)
16188: Procedure DebugRunClick( Sender : TObject)
16189: procedure Dec;
16190: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
16191: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
16192: Procedure DecodeDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
16193: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
16194: Procedure DecodeDateTime( const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
16195: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
16196: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
16197: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
16198: Procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
16199: Procedure DecompileClick( Sender : TObject)
16200: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
16201: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
16202: Procedure DeferLayout
16203: Procedure deffileread
16204: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
16205: Procedure DelayMicroseconds( const MicroSeconds : Integer)
16206: Procedure Delete
16207: Procedure Delete( const AFilename : string)
16208: Procedure Delete( const Index : Integer)
16209: Procedure DELETE( INDEX : INTEGER)
16210: Procedure Delete( Index : LongInt)
16211: Procedure Delete( Node : TTreeNode)
16212: procedure Delete(var s: AnyString; ifrom, icount: Longint);
16213: Procedure DeleteAlias( const Name : string)
16214: Procedure DeleteDriver( const Name : string)
16215: Procedure DeleteIndex( const Name : string)
16216: Procedure DeleteKey( const Section, Ident : String)
16217: Procedure DeleteRecords
16218: Procedure DeleteRecords( AffectRecords : TAffectRecords)
16219: Procedure DeleteString( var pStr : String; const pDelStr : string)
16220: Procedure DeleteTable
16221: procedure DelphiSiteClick(Sender: TObject);
16222: Procedure Deselect
16223: Procedure Deselect( Node : TTreeNode)
16224: procedure DestroyComponents
16225: Procedure DestroyHandle
16226: Procedure Diff( var X : array of Double)
16227: procedure Diff(var X: array of Double);
16228: procedure DISABLEALIGN
16229: Procedure DisableConstraints
16230: Procedure Disconnect
16231: Procedure Disconnect( Socket : TSocket)
16232: Procedure Dispose
16233: procedure Dispose(P: PChar)
16234: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
16235: Procedure DoKey( Key : TDBCtrlGridKey)
16236: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
16237: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
16238: Procedure Dormant
16239: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
16240: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
16241: Procedure DoubleToComp( Value : Double; var Result : Comp)
16242: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
16243: procedure Draw(X, Y: Integer; Graphic: TGraphic);
16244: Procedure Draw1(Canvas:TCanvas; X,Y,
Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
16245: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
16246: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
16247: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)

```

```

16248: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
16249: procedure DrawFocusRect(const Rect: TRect);
16250: Procedure DrawHDIbToTBitmap( HDIB : THandle; Bitmap : TBitmap)
16251: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCanvas; ARECT : TRect; STATE : TOWNERDRAWSTATE)
16252: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
16253: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
    TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
16254: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
16255: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
16256: Procedure DropConnections
16257: Procedure DropDown
16258: Procedure DumpDescription( oStrings : TStrings)
16259: Procedure DumpStateTable( oStrings : TStrings)
16260: Procedure EDIT
16261: Procedure EditButtonClick
16262: Procedure EditFont1Click( Sender : TObject)
16263: procedure Ellipse(X1, Y1, X2, Y2: Integer);
16264: Procedure Ellipse1( const Rect : TRect);
16265: Procedure EMMS
16266: Procedure Encode( ADest : TStream)
16267: procedure ENDDRAG(DROP:BOOLEAN)
16268: Procedure EndEdit( Cancel : Boolean)
16269: Procedure EndTimer
16270: Procedure EndUpdate
16271: Procedure EraseSection( const Section : string)
16272: Procedure Error( const Ident : string)
16273: procedure Error(Ident:Integer)
16274: Procedure ErrorFmt( const Ident : string; const Args : array of const)
16275: Procedure ErrorStr( const Message : string)
16276: procedure ErrorStr(Message:String)
16277: Procedure Exchange( Index1, Index2 : Integer)
16278: procedure Exchange(Index1, Index2: Integer);
16279: Procedure Exec( FileName, Parameters, Directory : string)
16280: Procedure ExecProc
16281: Procedure ExecSQL( UpdateKind : TUpdateKind)
16282: Procedure Execute
16283: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
16284: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
16285: Procedure ExecuteCommand(executeFile, paramstring: string)
16286: Procedure ExecuteShell(executeFile, paramstring: string)
16287: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
16288: Procedure ExitThread(ExitCode: Integer); stdcall;
16289: Procedure ExitProcess(ExitCode: Integer); stdcall;
16290: Procedure Expand( AUserName : string; AResults : TStrings)
16291: Procedure Expand( Recurse : Boolean)
16292: Procedure ExportClipboard1Click( Sender : TObject)
16293: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
16294: Procedure ExtractContentFields( Strings : TStrings)
16295: Procedure ExtractCookieFields( Strings : TStrings)
16296: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
16297: Procedure ExtractHeaderFields(Separ,
    WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuotes:Bool)
16298: Procedure ExtractHTTFFields(Separators, WhiteSpace:
    TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
16299: Procedure ExtractQueryFields( Strings : TStrings)
16300: Procedure FastDegToGrad
16301: Procedure FastDegToRad
16302: Procedure FastGradToDeg
16303: Procedure FastGradToRad
16304: Procedure FastRadToDeg
16305: Procedure FastRadToGrad
16306: Procedure FileClose( Handle : Integer)
16307: Procedure FileClose(handle: integer)
16308: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
16309: Procedure FileStructure( AStructure : TIdFTPDataStructure)
16310: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
16311: Procedure FillBytes( var VBytes: TIdBytes; const ACount : Integer; const AValue : Byte)
16312: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
16313: Procedure FillChar2(var X: PChar ; count: integer; value: char)
16314: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
16315: Procedure FillIPLList
16316: procedure FillRect(const Rect: TRect);
16317: Procedure FillTStrings( AStrings : TStrings)
16318: Procedure FilterOnBookmarks( Bookmarks : array of const)
16319: procedure FinalizePackage(Module: HMODULE)
16320: procedure FindClose;
16321: procedure FindClose2(var F: TSearchRec)
16322: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
16323: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
16324: Procedure FindNearest( const KeyValues : array of const)
16325: Procedure FinishContext
16326: Procedure FIRST
16327: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
16328: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
16329: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
16330: Procedure FlushSchemaCache( const TableName : string)
16331: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
16332: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
16333: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)

```

```

16334: Procedure FormActivate( Sender : TObject)
16335: procedure FormatLn(const format: String; const args: array of const); //alias
16336: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
16337: Procedure FormCreate( Sender : TObject)
16338: Procedure FormDestroy( Sender : TObject)
16339: Procedure FormKeyPress( Sender : TObject; var Key : Char)
16340: procedure FormOutputClick(Sender: TObject);
16341: Procedure FormToHtml( Form : TForm; Path : string)
16342: procedure FrameRect(const Rect: TRect);
16343: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
16344: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
16345: Procedure Free( Buffer : TRecordBuffer)
16346: Procedure Free( Buffer : TValueBuffer)
16347: Procedure Free;
16348: Procedure FreeAndNil(var Obj:TObject)
16349: Procedure FreeImage
16350: procedure FreeMem(P: PChar; Size: Integer)
16351: Procedure FreeTreeData( Tree : TUpdateTree)
16352: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
16353: Procedure FullCollapse
16354: Procedure FullExpand
16355: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short); //InterBase
16356: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
16357: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
16358: Procedure Get1( AURL : string; const AResponseContent : TStream);
16359: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
16360: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
16361: Procedure GetAliasNames( List : TStrings)
16362: Procedure GetAliasParams( const AliasName : string; List : TStrings)
16363: Procedure GetApplicationsRunning( Strings : TStrings)
16364: Procedure GetCommandTypes( List : TWideStrings)
16365: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
16366: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
16367: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
16368: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
16369: Procedure GetDatabaseNames( List : TStrings)
16370: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
16371: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
16372: Procedure GetDir(d: byte; var s: string)
16373: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
16374: Procedure GetDriverNames( List : TStrings)
16375: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
16376: Procedure GetDriverParams( const DriverName : string; List : TStrings)
16377: Procedure GetEMailsClick( Sender : TObject)
16378: Procedure getEnvironmentInfo;
16379: Function getEnvironmentString: string;
16380: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
16381: Procedure GetFieldNames( const TableName : string; List : TStrings)
16382: Procedure GetFieldNames( const TableName : string; List : TStrings);
16383: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
16384: Procedure GETFIELDNAMES( LIST : TSTRINGS)
16385: Procedure GetFieldNames1( const TableName : string; List : TStrings);
16386: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
16387: Procedure GetFieldNames2( const TableName : WideString; SchemaName : WideString; List : TWideStrings);
16388: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
16389: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
16390: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
16391: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
16392: Procedure GetFormatSettings
16393: Procedure GetFromDIB( var DIB : TBitmapInfo)
16394: Procedure GetFromHDIB( HDIB : HBitmap)
16395: Procedure GetIcon( Index : Integer; Image : TIcon);
16396: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
16397: Procedure GetIndexInfo( IndexName : string)
16398: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
16399: Procedure GetIndexNames( List : TStrings)
16400: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
16401: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
16402: Procedure GetIndexNames4( const TableName : string; List : TStrings);
16403: Procedure GetInternalResponse
16404: Procedure GETITEMNAMES( LIST : TSTRINGS)
16405: procedure GetMem(P: PChar; Size: Integer)
16406: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
16407: procedure GetPackageDescription(ModuleName: PChar): string)
16408: Procedure GetPackageNames( List : TStrings);
16409: Procedure GetPackageNames1( List : TWideStrings);
16410: Procedure GetParamList( List : TList; const ParamNames : WideString)
16411: Procedure GetProcedureNames( List : TStrings);
16412: Procedure GetProcedureNames( List : TWideStrings);
16413: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
16414: Procedure GetProcedureNames1( List : TStrings);
16415: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
16416: Procedure GetProcedureNames3( List : TWideStrings);
16417: Procedure GetProcedureNames4( const PackageName : WideString; List : TWideStrings);
16418: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
16419: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
16420: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
16421: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : WideString; List : TList);
16422: Procedure GetProviderNames( Names : TWideStrings);

```

```

16423: Procedure GetProviderNames( Proc : TGetStrProc)
16424: Procedure GetProviderNames1( Names : TStrings);
16425: procedure GetQRCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
16426: procedure GetQRCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no autoopen
image
16427: Function GetQRCode4(Width,Height:Word;Correct_Level:string; const
Data:string;aformat:string):TLinearBitmap;
16428: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
16429: Procedure GetSchemaNames( List : TStrings);
16430: Procedure GetSchemaNames1( List : TWideStrings);
16431: Procedure GetSessionNames( List : TStrings)
16432: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
16433: Procedure GetStrings( List : TStrings)
16434: Procedure GetSystemTime; stdcall;
16435: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
16436: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
16437: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
16438: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
16439: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
16440: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
16441: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
16442: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
16443: Procedure GetVisibleWindows( List : Tstrings)
16444: Procedure GoBegin
16445: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
16446: Procedure GotoCurrent( Table : TTable)
16447: procedure GotoEndClick(Sender: TObject);
16448: Procedure GotoNearest
16449: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
Direction: TGradientDirection)
16450: Procedure HandleException( E : Exception; var Handled : Boolean)
16451: procedure HANDLEMESSAGE
16452: procedure HandleNeeded;
16453: Procedure Head( AURL : string)
16454: Procedure Help( var AHelpContents : TStringList; ACommand : string)
16455: Procedure HexToBinary( Stream : TStream)
16456: procedure HexToBinary(Stream:TStream)
16457: Procedure HideDragImage
16458: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
16459: Procedure HideTraybar
16460: Procedure HideWindowForSeconds(secs: integer); { //3 seconds}
16461: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); { //3 seconds}
16462: Procedure HookOSExceptions
16463: Procedure HookSignal( RtlSigNum : Integer)
16464: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
16465: Procedure HTMLSyntax1Click( Sender : TObject)
16466: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
16467: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
16468: Procedure ImportfromClipboard1Click( Sender : TObject)
16469: Procedure ImportfromClipboard2Click( Sender : TObject)
16470: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
16471: procedure Incb(var x: byte);
16472: Procedure Include1Click( Sender : TObject)
16473: Procedure IncludeOFF; //preprocessing
16474: Procedure IncludeON;
16475: procedure Info1Click(Sender: TObject);
16476: Procedure InitAltRecBuffers( CheckModified : Boolean)
16477: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
16478: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
16479: Procedure InitData( ASource : TDataSet)
16480: Procedure InitDelta( ADelta : TPacketDataSet);
16481: Procedure InitDelta( const ADelta : OleVariant);
16482: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
16483: Procedure Initialize
16484: procedure InitializePackage(Module: HMODULE)
16485: Procedure INITIATEACTION
16486: Procedure initHexArray(var hexn: THexArray); //THexArray', 'array[0..15] of char;
16487: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
16488: Procedure InitModule( AModule : TComponent)
16489: Procedure InitStdConvs
16490: Procedure InitTreeData( Tree : TUpdateTree)
16491: Procedure INSERT
16492: Procedure Insert( Index : Integer; AClass : TClass)
16493: Procedure Insert( Index : Integer; AComponent : TComponent)
16494: Procedure Insert( Index : Integer; AObject : TObject)
16495: Procedure Insert( Index : Integer; const S : WideString)
16496: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
16497: Procedure Insert(Index: Integer; const S: string);
16498: procedure Insert(Index: Integer; S: string);
16499: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
16500: procedure InsertComponent(AComponent:TComponent)
16501: procedure InsertControl(AControl: TControl);
16502: Procedure InsertIcon( Index : Integer; Image : TIcon)
16503: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
16504: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
16505: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
16506: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
16507: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
16508: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)

```



```

16509: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
16510: Procedure InternalBeforeResolve( Tree : TUpdateTree)
16511: Procedure InvalidateModuleCache
16512: Procedure InvalidateTitles
16513: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
16514: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
16515: Procedure InvalidDateTimeError(const AYear, AMth, ADay, AHour, AMin, ASec, AMilSec: Word; const
  ABaseDate: TDateTime)
16516: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
16517: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
16518: procedure JavaSyntax1Click(Sender: TObject);
16519: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
16520: Procedure KillDataChannel
16521: Procedure Largefont1Click( Sender : TObject)
16522: Procedure LAST
16523: Procedure LaunchCpl( FileName : string)
16524: Procedure Launch( const AFile : string)
16525: Procedure LaunchFile( const AFile : string)
16526: Procedure LetFileList(FileList: TStringlist; apath: string);
16527: Procedure lineToNumber( xmemo : String; met : boolean)
16528: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
  DefaultDraw:Bool)
16529: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
  : TCustomDrawState; var DefaultDraw : Boolean)
16530: Procedure ListViewData( Sender : TObject; Item : TListItem)
16531: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
  : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
16532: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
16533: Procedure ListViewDblClick( Sender : TObject)
16534: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
16535: Procedure ListDLLExports(const FileName: string; List: TStrings);
16536: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
16537: procedure LoadBytecode1Click(Sender: TObject);
16538: procedure LoadFileFromResource(const FileName: string; ms: TMemoryStream);
16539: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
16540: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
16541: Procedure LoadFromFile( AFileName : string)
16542: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
16543: Procedure LoadFromFile( const FileName : string)
16544: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
16545: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
16546: Procedure LoadFromFile( const FileName : WideString)
16547: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
16548: Procedure LoadFromFile(const AFileName: string)
16549: procedure LoadFromFile(FileName: string);
16550: procedure LoadFromFile(FileName:String)
16551: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
16552: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
16553: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
16554: Procedure LoadFromStream( const Stream : TStream)
16555: Procedure LoadFromStream( S : TStream)
16556: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16557: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16558: Procedure LoadFromStream( Stream : TStream)
16559: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
16560: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
16561: procedure LoadFromStream(Stream: TStream);
16562: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
16563: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
16564: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
16565: Procedure LoadLastFile1Click( Sender : TObject)
16566: { LoadIcoToImage loads two icons from resource named NameRes,
  into two image lists ALarge and ASmall}
16567:
16568: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
16569: Procedure LoadMemo
16570: Procedure LoadParamsFromIniFile( FFileName : WideString)
16571: Procedure Lock
16572: Procedure Login
16573: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
16574: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
16575: Procedure MakeCaseInsensitive
16576: Procedure MakeDeterministic( var bChanged : boolean)
16577: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
16578: // type TVolumeLevel = 0..127; , saveFilePath as C:\MyFile.wav
16579: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
16580: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
16581: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
16582: Procedure SetRectComplexFormatStr( const S : string)
16583: Procedure SetPolarComplexFormatStr( const S : string)
16584: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
16585: Procedure MakeVisible
16586: Procedure MakeVisible( PartialOK : Boolean)
16587: Procedure Manual1Click( Sender : TObject)
16588: Procedure MarkReachable
16589: Procedure maXbox; //shows the exe version data in a win box
16590: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
16591: Procedure Memo1Change( Sender : TObject)
16592: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
  Action:TSynReplaceAction)

```

```

16593: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
16594: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
16595: procedure Memory1Click(Sender: TObject);
16596: Procedure MERGE( MENU : TMAINMENU)
16597: Procedure MergeChangeLog
16598: procedure MINIMIZE
16599: Procedure MinimizeMaxbox;
16600: Procedure MkDir(const s: string)
16601: Procedure mnuPrintFont1Click( Sender : TObject)
16602: procedure ModalStarted
16603: Procedure Modified
16604: Procedure ModifyAlias( Name : string; List : TStrings)
16605: Procedure ModifyDriver( Name : string; List : TStrings)
16606: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
16607: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
16608: Procedure Move( CurIndex, NewIndex : Integer)
16609: procedure Move(CurIndex, NewIndex: Integer);
16610: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
16611: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
16612: Procedure moveCube( o : TMyLabel)
16613: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
16614: procedure MoveTo(X, Y: Integer);
16615: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
16616: Procedure MovePoint(var x,y:Extended; const angle:Extended);
16617: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
16618: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
16619: Procedure MsgAbout( Handle: Int;const Msg,Caption:string;const IcoName:string = 'MAINICON';Flags:DWORD=MB_OK);
16620: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
16621: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
16622: procedure New(P: PChar)
16623: procedure New1Click(Sender: TObject);
16624: procedure NewInstance1Click(Sender: TObject);
16625: Procedure NEXT
16626: Procedure NextMonth
16627: Procedure Noop
16628: Procedure NormalizePath( var APath : string)
16629: procedure ObjectBinaryToText(Input, Output: TStream)
16630: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16631: procedure ObjectResourceToText(Input, Output: TStream)
16632: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16633: procedure ObjectTextToBinary(Input, Output: TStream)
16634: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16635: procedure ObjectTextToResource(Input, Output: TStream)
16636: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16637: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
16638: Procedure Open( const UserID : WideString; const Password : WideString);
16639: Procedure Open;
16640: Procedure open1Click( Sender : TObject)
16641: Procedure OpenCdDrive
16642: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
16643: Procedure OpenCurrent
16644: Procedure OpenFile(vfilenamepath: string)
16645: Procedure OpenDirectory1Click( Sender : TObject)
16646: Procedure OpenIndexFile( const IndexName : string)
16647: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
SchemaID:OleVariant;DataSet:TADODataSet)
16648: Procedure OpenWriteBuffer( const AThreshold : Integer)
16649: Procedure OptimizeMem
16650: Procedure Options1( AURL : string);
16651: Procedure OutputDebugString(lpOutputString : PChar)
16652: Procedure PackBuffer
16653: Procedure Paint
16654: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
16655: Procedure PaintToTBitmap( Target : TBitmap)
16656: Procedure PaletteChanged
16657: Procedure ParentBiDiModeChanged
16658: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
16659: Procedure PasteFromClipboard;
16660: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
16661: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
16662: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
16663: Procedure PError( Text : string)
16664: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
16665: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Integer;Y3:Integer;X4:Integer;Y4:Integer);
16666: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
16667: procedure playmp3(mpath: string);
16668: Procedure PlayMP31Click( Sender : TObject)
16669: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
16670: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
16671: procedure PolyBezier(const Points: array of TPoint);
16672: procedure PolyBezierTo(const Points: array of TPoint);
16673: procedure Polygon(const Points: array of TPoint);
16674: procedure Polyline(const Points: array of TPoint);
16675: Procedure Pop
16676: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
16677: Procedure PopulationVarianceAndMean( const X : TdynFloatArray; var Variance, Mean : Float)
16678: Procedure POPUP( X, Y : INTEGER)
16679: Procedure PopupURL(URL : WideString);
16680: Procedure POST

```

```

16681: Procedure Post4( AURL : string; const ASource : TString; const AResponseContent : TStream);
16682: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
16683: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
16684: Procedure PostUser( const Email, FirstName, LastName : WideString);
16685: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
16686: procedure Pred(X: int64);
16687: Procedure Prepare
16688: Procedure PrepareStatement
16689: Procedure PreProcessXML( AList : TString)
16690: Procedure PreventDestruction
16691: Procedure Print( const Caption : string)
16692: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
16693: procedure printf(const format: string; const args: array of const);
16694: Procedure PrintList(Value: TStringList);
16695: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
16696: Procedure Printout1Click( Sender : TObject)
16697: Procedure ProcessHeaders
16698: Procedure PROCESSMENUCHAR( var MESSAGE : TWMENUCHAR)
16699: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
16700: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
16701: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
16702: Procedure ProcessMessagesOFF; //application.processmessages
16703: Procedure ProcessMessagesON;
16704: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
16705: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
16706: Procedure Proclst Size is: 3797 /1415
16707: Procedure procMessClick( Sender : TObject)
16708: Procedure PSScriptCompile( Sender : TPSScript)
16709: Procedure PSScriptExecute( Sender : TPSScript)
16710: Procedure PSScriptLine( Sender : TObject)
16711: Procedure Push( ABoundary : string)
16712: procedure PushItem(AItem: Pointer)
16713: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
16714: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
16715: procedure PutLinuxLines(const Value: string)
16716: Procedure Quit
16717: Procedure RaiseConversionError( const AText : string);
16718: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
16719: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
16720: procedure RaiseException(Ex: TIFException; Param: string);
16721: Procedure RaiseExceptionForLastCmdResult;
16722: procedure RaiseLastException;
16723: procedure RaiseException2;
16724: Procedure RaiseLastOSError
16725: Procedure RaiseLastWin32;
16726: procedure RaiseLastWin32Error)
16727: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
16728: Procedure RandomFillStream( Stream : TMemoryStream)
16729: procedure randomize;
16730: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
16731: Procedure RCS
16732: Procedure Read( Socket : TSocket)
16733: Procedure ReadBlobData
16734: procedure ReadBuffer(Buffer:string;Count:LongInt)
16735: procedure ReadOnly1Click(Sender: TObject);
16736: Procedure ReadSection( const Section : string; Strings : TString)
16737: Procedure ReadSections( Strings : TString)
16738: Procedure ReadSections( Strings : TString);
16739: Procedure ReadSections1( const Section : string; Strings : TString);
16740: Procedure ReadSectionValues( const Section : string; Strings : TString)
16741: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
16742: Procedure ReadStrings( ADest : TString; AReadLinesCount : Integer)
16743: Procedure ReadVersion2(aFileName: STRING; aVersion : TString);
16744: Function ReadVersion(aFileName: STRING; aVersion : TString): boolean;
16745: Procedure Realign;
16746: procedure Rectangle(X1, Y1, X2, Y2: Integer);
16747: Procedure Rectangle1( const Rect : TRect);
16748: Procedure RectCopy( var Dest : TRect; const Source : TRect)
16749: Procedure RectFitToScreen( var R : TRect)
16750: Procedure RectGrow( var R : TRect; const Delta : Integer)
16751: Procedure RectGrowX( var R : TRect; const Delta : Integer)
16752: Procedure RectGrowY( var R : TRect; const Delta : Integer)
16753: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
16754: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
16755: Procedure RectNormalize( var R : TRect)
16756: // TFileCallbackProcedure = procedure(filename:string);
16757: Procedure RecurseDirectory(Dir: string;IncludeSubs: boolean;callback: TFileCallbackProcedure);
16758: Procedure RecurseDirectory2(Dir: string; IncludeSubs : boolean);
16759: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
16760: Procedure Refresh;
16761: Procedure RefreshData( Options : TFetchOptions)
16762: Procedure REFRESHLOOKUPLIST
16763: Procedure RegisterAuthenticationMethod( MethodName : string; AuthClass : TIdAuthenticationClass)
16764: Procedure RegisterChanges( Value : TChangeLink)
16765: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
16766: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
16767: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
16768: Procedure ReInitialize( ADelay : Cardinal)
16769: procedure RELEASE

```

```

16770: Procedure Remove( const AByteCount : integer)
16771: Procedure REMOVE( FIELD : TFIELD)
16772: Procedure REMOVE( ITEM : TMenuItem)
16773: Procedure REMOVE( POPUP : TPopupMenu)
16774: Procedure RemoveAllPasswords
16775: procedure RemoveComponent(AComponent:TComponent)
16776: Procedure RemoveDir( const ADirName : string)
16777: Procedure RemoveLambdaTransitions( var bChanged : boolean)
16778: Procedure REMOVEPARAM( VALUE : TPARAM)
16779: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
16780: Procedure RemoveTransitionTo( oState : TniRegularExpressionState);
16781: Procedure Rename( const ASourceFile, ADestFile : string)
16782: Procedure Rename( const FileName : string; Reload : Boolean)
16783: Procedure RenameTable( const NewTableName : string)
16784: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
16785: Procedure ReplaceClick( Sender : TObject)
16786: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
16787: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
16788: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
16789: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
16790: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
16791: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
16792: Procedure Requery( Options : TExecuteOptions)
16793: Procedure Reset
16794: Procedure ResetClick( Sender : TObject)
16795: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
16796: procedure ResourceExploreClick(Sender: TObject);
16797: Procedure RestoreContents
16798: Procedure RestoreDefaults
16799: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
16800: Procedure RetrieveHeaders
16801: Procedure RevertRecord
16802: Procedure RGBAToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
16803: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16804: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16805: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
16806: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
16807: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
16808: Procedure RleCompress2( Stream : TStream)
16809: Procedure RleDecompress2( Stream : TStream)
16810: Procedure Rmdir(const S: string)
16811: Procedure Rollback
16812: Procedure Rollback( TransDesc : TTransactionDesc)
16813: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
16814: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
16815: Procedure RollbackTrans
16816: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
16817: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
16818: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
16819: Procedure RunDll32Internal( Wnd : HWND; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
16820: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
16821: Procedure S_EBox( const AText : string)
16822: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
AddKey:int
16823: Procedure S_IBox( const AText : string)
16824: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
16825: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
16826: Procedure S-TokenInit( cBuffer : PChar; const cDelimiters : string)
16827: Procedure SampleVarianceAndMean
16828: ( const X : TDynFloatArray; var Variance, Mean : Float)
16829: Procedure Save2Click( Sender : TObject)
16830: Procedure Saves3Click( Sender : TObject)
16831: Procedure SavebeforeClick( Sender : TObject)
16832: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
16833: Procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
16834: Procedure SaveConfigFile
16835: Procedure SaveOutputClick( Sender : TObject)
16836: procedure SaveScreenshotClick(Sender: TObject);
16837: Procedure SaveLn(pathname, content: string); //SaveLn(exepath+'mysaveIntest.txt', memo2.text);
16838: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
16839: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
16840: Procedure SaveToFile( AFileName : string)
16841: Procedure SAVETOFILE( const FILENAME : String)
16842: Procedure SaveToFile( const FileName : WideString)
16843: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
16844: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
16845: procedure SaveToFile(FileName: string);
16846: procedure SaveToFile(FileName:String)
16847: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
16848: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16849: Procedure SaveToStream( S : TStream)
16850: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16851: Procedure SaveToStream( Stream : TStream)
16852: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
16853: procedure SaveToStream(Stream: TStream);
16854: procedure SaveToStream(Stream:TStream)
16855: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
16856: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
16857: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)

```



```

16858: procedure Say(const sText: string)
16859: Procedure SBytecodeClick( Sender : TObject)
16860: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
16861: procedure ScriptExplorerClick(Sender: TObject);
16862: Procedure Scroll( Distance : Integer)
16863: Procedure Scroll( DX, DY : Integer)
16864: procedure ScrollBy(DeltaX, DeltaY: Integer);
16865: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
16866: Procedure ScrollTabs( Delta : Integer)
16867: Procedure Search1Click( Sender : TObject)
16868: procedure SearchAndOpenDoc(vfilenamepath: string)
16869: procedure SearchAndOpenFile(vfilenamepath: string)
16870: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
16871: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
16872: Procedure SearchNext1Click( Sender : TObject)
16873: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
16874: Procedure Select1( const Nodes : array of TTreeNode);
16875: Procedure Select2( Nodes : TList);
16876: Procedure SelectNext( Direction : Boolean)
16877: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
16878: Procedure SelfTestPEM //unit uPSI_cPEM
16879: Procedure Send( AMsg : TIdMessage)
16880: //config forst in const MAILINIFILE = 'maildef.ini';
16881: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox',
16882: Procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
16883: Procedure SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
16884: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
16885: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
16886: Procedure SendResponse
16887: Procedure SendStream( AStream : TStream)
16888: Procedure Set8087CW( NewCW : Word)
16889: Procedure SetAll( One, Two, Three, Four : Byte)
16890: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
16891: Procedure SetAppDispatcher( const ADispatcher : TComponent)
16892: procedure SetArrayLength;
16893: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer); //2 dimension
16894: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
16895: Procedure SetAsHandle( Format : Word; Value : THandle)
16896: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
16897: procedure SetCaptureControl(Control: TControl);
16898: Procedure SetColumnAttributes
16899: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
16900: Procedure SetCustomHeader( const Name, Value : string)
16901: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
FieldNames:Widestring)
16902: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
16903: Procedure SetFocus
16904: procedure SetFocus; virtual;
16905: Procedure SetInitialState
16906: Procedure SetKey
16907: procedure SetLastError(ErrorCode: Integer)
16908: procedure SetLength;
16909: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
16910: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
16911: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
16912: Procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
16913: Procedure SetParams1( UpdateKind : TUpdateKind);
16914: Procedure SetPassword( const Password : string)
16915: Procedure SetPointer( Ptr : Pointer; Size : Longint)
16916: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
16917: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
16918: Procedure SetProvider( Provider : TComponent)
16919: Procedure SetProxy( const Proxy : string)
16920: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
16921: Procedure SetRange( const StartValues, EndValues : array of const)
16922: Procedure SetRangeEnd
16923: Procedure SetRate( const aPercent, aYear : integer)
16924: Procedure SetRate(const aPercent, aYear: integer)
16925: Procedure Set_ReportMemoryLeaksOnShutdown(ab0: boolean)
16926: Procedure SetSafeCallExceptionMsg( Msg : String)
16927: procedure SETSELTEXTBUF(BUFFER:PCHAR)
16928: Procedure SetSize( AWidth, AHeight : Integer)
16929: procedure SetSize(NewSize:LongInt)
16930: procedure SetString(var s: string; buffer: PChar; len: Integer)
16931: Procedure SetStrings( List : TStrings)
16932: Procedure SetText( Text : PwideChar)
16933: procedure SetText(Text: PChar);
16934: Procedure SetTextBuf( Buffer : PChar)
16935: procedure SETTEXTBUF(BUFFER:PCHAR)
16936: Procedure SetTick( Value : Integer)
16937: Procedure SetTimeout( ATimeOut : Integer)
16938: Procedure SetTraceEvent( Event : TDBXTraceEvent)
16939: Procedure SetUserName( const UserName : string)
16940: Procedure SetWallpaper( Path : string);
16941: procedure ShellStyle1Click(Sender: TObject);
16942: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
16943: Procedure ShowFileProperties( const FileName : string)
16944: Procedure ShowInclude1Click( Sender : TObject)
16945: Procedure ShowInterfaces1Click( Sender : TObject)

```

```

16946: Procedure ShowLastException1Click( Sender : TObject)
16947: Procedure ShowMessage( const Msg : string)
16948: Procedure ShowMessageBig(const aText : string);
16949: Procedure ShowMessageBig2(const aText : string; autosize: boolean);
16950: Procedure ShowMessageBig3(const aText : string; fsize: byte; autosize: boolean);
16951: Procedure MsgBig(const aText : string); //alias
16952: procedure showmessage(mytext: string);
16953: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
16954: procedure ShowMessageFmt(const Msg: string; Params: array of const))
16955: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
16956: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
16957: Procedure ShowSearchDialog( const Directory : string)
16958: Procedure ShowSpecChars1Click( Sender : TObject)
16959: Procedure ShowBitmap(bmap: TBitmap); //draw in a form!
16960: Procedure ShredFile( const FileName : string; Times : Integer)
16961: procedure Shuffle(vQ: TStringList);
16962: Procedure ShuffleList( var List : array of Integer; Count : Integer)
16963: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
16964: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
16965: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
16966: Procedure Site( const ACommand : string)
16967: Procedure SkipEOL
16968: Procedure Sleep( ATime : cardinal)
16969: Procedure Sleep( milliseconds : Cardinal)
16970: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
16971: Procedure Slinenumber1Click( Sender : TObject)
16972: Procedure Sort
16973: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
16974: procedure Speak(const sText: string) //async like voice
16975: procedure Speak2(const sText: string) //sync
16976: procedure Split(Str: string; SubStr: string; List: TStringList);
16977: Procedure SplitNameValue( const Line : string; var Name, Value : string)
16978: Procedure SplitColumns( const AData : string; AStrings : TStringList; const ADelim : string)
16979: Procedure SplitColumnsNoTrim( const AData : string; AStrings : TStringList; const ADelim : string)
16980: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStringList)
16981: Procedure SplitString( const AStr, AToken : string; var VLeft, VRight : string)
16982: procedure SQLSyntax1Click(Sender: TObject);
16983: Procedure SRand( Seed : RNG_IntType)
16984: Procedure Start
16985: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
16986: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringList);
16987: Procedure StartTransaction( TransDesc : TTransactionDesc)
16988: Procedure Status( var AStatusList : TStringList)
16989: Procedure StatusBar1Db1Click( Sender : TObject)
16990: Procedure StepInto1Click( Sender : TObject)
16991: Procedure StepIt
16992: Procedure StepOut1Click( Sender : TObject)
16993: Procedure Stop
16994: procedure stopmp3;
16995: procedure StartWeb(aurl: string);
16996: Procedure Str(aint: integer; astr: string); //of system
16997: Procedure StrDispose( Str : PChar)
16998: procedure StrDispose(Str: PChar)
16999: Procedure StrReplace(var Str: string; Old, New: string);
17000: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
17001: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
17002: Procedure StringToBytes( Value : string; Bytes : array of byte)
17003: procedure StrSet(c : Char; I : Integer; var s : string);
17004: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStringList);
17005: Procedure StructureMount( APath : string)
17006: procedure STYLECHANGED(SENDER:TObJECT)
17007: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
17008: procedure Succ(X: int64);
17009: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
17010: procedure SwapChar(var X,Y: char); //swapX follows
17011: Procedure SwapFloats( var X, Y : Float)
17012: procedure SwapGrid(grd: TStringGrid);
17013: Procedure SwapOrd( var I, J : Byte)
17014: Procedure SwapOrd( var X, Y : Integer)
17015: Procedure SwapOrd1( var I, J : Shortint);
17016: Procedure SwapOrd2( var I, J : Smallint);
17017: Procedure SwapOrd3( var I, J : Word);
17018: Procedure SwapOrd4( var I, J : Integer);
17019: Procedure SwapOrd5( var I, J : Cardinal);
17020: Procedure SwapOrd6( var I, J : Int64);
17021: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
17022: Procedure Synchronizel( Method : TMethod);
17023: procedure SyntaxCheck1Click(Sender: TObject);
17024: Procedure SysFreeString(const S: WideString); stdcall;
17025: Procedure TakeOver( Other : TLinearBitmap)
17026: Procedure TalkIn(const sText: string) //async voice
17027: procedure tbtn6resClick(Sender: TObject);
17028: Procedure tbtnUseCaseClick( Sender : TObject)
17029: procedure TerminalStyle1Click(Sender: TObject);
17030: Procedure Terminate
17031: Procedure texSyntax1Click( Sender : TObject)
17032: procedure TextOut(X, Y: Integer; Text: string);
17033: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
17034: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);

```

```

17035: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
17036: Procedure TextStart
17037: procedure TILE
17038: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
17039: Procedure TitleClick( Column : TColumn)
17040: Procedure ToDo
17041: procedure toolbtnTutorialClick(Sender: TObject);
17042: Procedure Trace1( AURL : string; const AResponseContent : TStream);
17043: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
17044: Procedure Truncate
17045: procedure Tutorial101Click(Sender: TObject);
17046: procedure Tutorial10Statistics1Click(Sender: TObject);
17047: procedure Tutorial11Forms1Click(Sender: TObject);
17048: procedure Tutorial12SQL1Click(Sender: TObject);
17049: Procedure tutorial1Click( Sender : TObject)
17050: Procedure tutorial21Click( Sender : TObject)
17051: Procedure tutorial31Click( Sender : TObject)
17052: Procedure tutorial4Click( Sender : TObject)
17053: Procedure Tutorial5Click( Sender : TObject)
17054: procedure Tutorial6Click(Sender: TObject);
17055: procedure Tutorial91Click(Sender: TObject);
17056: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
17057: procedure UniqueString(var str: AnsiString)
17058: procedure UnloadLoadPackage(Module: HMODULE)
17059: Procedure Unlock
17060: Procedure UNMERGE( MENU : TMAINMENU)
17061: Procedure UnRegisterChanges( Value : TChangeLink)
17062: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
17063: Procedure UnregisterConversionType( const AType : TConvType)
17064: Procedure UnRegisterProvider( Prov : TCustomProvider)
17065: Procedure UPDATE
17066: Procedure UpdateBatch( AffectRecords : TAffectRecords)
17067: Procedure UPDATECURSORPOS
17068: Procedure UpdateFile
17069: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
17070: Procedure UpdateResponse( AResponse : TWebResponse)
17071: Procedure UpdateScrollBar
17072: Procedure UpdateView1Click( Sender : TObject)
17073: procedure Val(const s: string; var n, z: Integer)
17074: procedure VarArraySet(c: Variant; I: Integer; var s: Variant);
17075: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
17076: Procedure VariantAdd( const src : Variant; var dst : Variant)
17077: Procedure VariantAnd( const src : Variant; var dst : Variant)
17078: Procedure VariantArrayRedim( var V : Variant; High : Integer)
17079: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
17080: Procedure VariantClear( var V : Variant)
17081: Procedure VariantCpy( const src : Variant; var dst : Variant)
17082: Procedure VariantDiv( const src : Variant; var dst : Variant)
17083: Procedure VariantMod( const src : Variant; var dst : Variant)
17084: Procedure VariantMul( const src : Variant; var dst : Variant)
17085: Procedure VariantOr( const src : Variant; var dst : Variant)
17086: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
17087: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
17088: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
17089: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
17090: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
17091: Procedure VariantShl( const src : Variant; var dst : Variant)
17092: Procedure VariantShr( const src : Variant; var dst : Variant)
17093: Procedure VariantSub( const src : Variant; var dst : Variant)
17094: Procedure VariantXor( const src : Variant; var dst : Variant)
17095: Procedure VarCastError;
17096: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
17097: Procedure VarInvalidOp
17098: Procedure VarInvalidNullOp
17099: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
17100: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
17101: Procedure VarArrayCreateError
17102: Procedure VarResultCheck( AResult : HRESULT);
17103: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
17104: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
17105: Function VarTypeAsText( const AType : TVarType) : string
17106: procedure Voice(const sText: string) //async
17107: procedure Voice2(const sText: string) //sync
17108: Procedure WaitMiliSeconds( AMSec : word)
17109: Procedure WideAppend( var dst : WideString; const src : WideString)
17110: Procedure WideAssign( var dst : WideString; var src : WideString)
17111: Procedure WideDelete( var dst : WideString; index, count : Integer)
17112: Procedure WideFree( var s : WideString)
17113: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
17114: Procedure WideFromPChar( var dst : WideString; src : PChar)
17115: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
17116: Procedure WideSetLength( var dst : WideString; len : Integer)
17117: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
17118: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
17119: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
17120: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
17121: Procedure HttpGet(const Url: string; Stream:TStream);
17122: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
17123: Procedure WordWrap1Click( Sender : TObject)

```

```

17124: Procedure Write( const AOut : string)
17125: Procedure Write( Socket : TSocket)
17126: procedure Write(S: string);
17127: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
17128: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
17129: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
17130: procedure WriteBuffer(Buffer:String;Count:LongInt)
17131: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
17132: Procedure WriteChar( AValue : Char)
17133: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
17134: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
17135: Procedure WriteFloat( const Section, Name : string; Value : Double)
17136: Procedure WriteHeader( AHeader : TStrings)
17137: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
17138: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
17139: Procedure WriteLn( const AOut : string)
17140: procedure Writeln(s: string);
17141: Procedure WriteLog( const FileName, LogLine : string)
17142: Procedure WriteRFCReply( AReply : TIdRFCReply)
17143: Procedure WriteRFCStrings( AStrings : TStrings)
17144: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
17145: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
17146: Procedure WriteString( const Section, Ident, Value : string)
17147: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
17148: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
17149: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
17150: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
17151: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
17152: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
17153: procedure XMLSyntax1Click(Sender: TObject);
17154: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
17155: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
17156: Procedure ZeroFillStream( Stream : TMemoryStream)
17157: procedure XMLSyntax1Click(Sender: TObject);
17158: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
17159: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
17160: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
17161: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
17162: procedure(Sender, Source: TObject; X, Y: Integer)
17163: procedure(Sender, Target: TObject; X, Y: Integer)
17164: procedure(Sender: TObject; ASection, AWidth: Integer)
17165: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
17166: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
17167: procedure(Sender: TObject; var Action: TCloseAction)
17168: procedure(Sender: TObject; var CanClose: Boolean)
17169: procedure(Sender: TObject; var Key: Char);
17170: ProcedureName ProcedureNames ProcedureParametersCursor @
17171:
17172: *****Now Constructors constructor *****
17173: Size is: 1248 1115 996 628 550 544 501 459 (381)
17174: Attach( VersionInfoData : Pointer; Size : Integer)
17175: constructor Create( ABuckets : TBucketListSizes)
17176: Create( ACallbackWnd : HWND)
17177: Create( AClient : TCustomTaskDialog)
17178: Create( AClient : TIdTelnet)
17179: Create( ACollection : TCollection)
17180: Create( ACollection : TFavoriteLinkItems)
17181: Create( ACollection : TTaskDialogButtons)
17182: Create( AConnection : TIdCustomHTTP)
17183: Create( ACreateSuspended : Boolean)
17184: Create( ADataSet : TCustomSQLDataSet)
17185: CREATE( ADATASET : TDATASET)
17186: Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
17187: Create( AGrid : TCustomDBGrid)
17188: Create( AGrid : TStringGrid; AIndex : Longint)
17189: Create( AHTTP : TIdCustomHTTP)
17190: Create( AListItems : TListItems)
17191: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
17192: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
17193: Create( AOwner : TCommonCalendar)
17194: Create( AOwner : TComponent)
17195: CREATE( AOWNER : TCOMPONENT)
17196: Create( AOwner : TCustomListView)
17197: Create( AOwner : TCustomOutline)
17198: Create( AOwner : TCustomRichEdit)
17199: Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
17200: Create( AOwner : TCustomTreeView)
17201: Create( AOwner : TIdUserManager)
17202: Create( AOwner : TListItems)
17203:
17204: Create(AOwner:TObject;Handl:hDBICur;CBType:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbckEvent:TBDECallbackEvent;Chain:Bool)
17204: CREATE( AOWNER : TPERSISTENT)
17205: Create( AOwner : TPersistent)
17206: Create( AOwner : TTable)
17207: Create( AOwner : TTreeNode)
17208: Create( AOwner : TWinControl; const ClassName : string)
17209: Create( AParent : TIdCustomHTTP)
17210: Create( AParent : TUpdateTree; AResolver : TCustomResolver)
17211: Create( AProvider : TBaseProvider)

```



```

17212: Create( AProvider : TCustomProvider);
17213: Create( AProvider : TDataSetProvider)
17214: Create( ASocket : TCustomWinSocket; TimeOut : Longint)
17215: Create( ASocket : TSocket)
17216: Create( AStrings : TWideStrings)
17217: Create( AToolBar : TToolBar)
17218: Create( ATreeNodes : TTreeNode)
17219: Create( Autofill : boolean)
17220: Create( AWebPageInfo : TAbstractWebPageInfo)
17221: Create( AWebRequest : TWebRequest)
17222: Create( Collection : TCollection)
17223: Create( Collection : TIdMessageParts; ABody : TStrings)
17224: Create( Collection : TIdMessageParts; const AFileName : TFileName)
17225: Create( Column : TColumn)
17226: Create( const AConvFamily : TConvFamily; const ADescription : string)
17227: Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
17228: Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
  AFromCommonProc : TConversionProc)
17229: Create( const AInitialState : Boolean; const AManualReset : Boolean)
17230: Create( const ATabSet : TTabSet)
17231: Create( const Compensate : Boolean)
17232: Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
17233: Create( const FileName : string)
17234: Create( const FileName : string; FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
  : Int64; const SecAttr : PSecurityAttributes);
17235: Create( const FileName : string; FileMode : WordfmShareDenyWrite)
17236: Create( const MaskValue : string)
17237: Create( const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
17238: Create( const Prefix : string)
17239: Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
17240: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
17241: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
17242: Create( CoolBar : TCoolBar)
17243: Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
17244: Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
17245: Create( DataSet :TDataSet; const
  Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
  DepFields : TBits; FieldMap : TFieldMap)
17246: Create( DBCtrlGrid : TDBCtrlGrid)
17247: Create( DSTableProducer : TDSTableProducer)
17248: Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
17249: Create( ErrorCode : DBIResult)
17250: Create( Field : TBlobField; Mode : TBlobStreamMode)
17251: Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
17252: Create( HeaderControl : TCustomHeaderControl)
17253: Create( HTTPRequest : TWebRequest)
17254: Create( iStart : integer; sText : string)
17255: Create( iValue : Integer)
17256: Create( Kind : TmmTimerKind; Notification : TmmNotificationKind)
17257: Create( MciErrNo : MCIERROR; const Msg : string)
17258: Create( MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
17259: Create( Message : string; ErrorCode : DBResult)
17260: Create( Msg : string)
17261: Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
17262: Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
17263: Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
17264: Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
17265: Create( oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
17266: Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
17267: Create( Owner : TCustomComboBoxEx)
17268: CREATE( OWNER : TINDEXTDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXTOPTIONS)
17269: Create( Owner : TPersistent)
17270: Create( Params : TStrings)
17271: Create( Size : Cardinal)
17272: Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
17273: Create( StatusBar : TCustomStatusBar)
17274: Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
17275: Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
17276: Create( AHandle:Integer)
17277: Create( AOwner: TComponent); virtual;
17278: Create( const AURI : string)
17279: Create( FileName:String;Mode:Word)
17280: Create( Instance:THandle;ResName:String;ResType:PChar)
17281: Create( Stream : TStream)
17282: Create( ADataset : TDataset);
17283: Create( const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
  SecAttr:PSecurityAttributes);
17284: Create( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
17285: Create2( Other : TObject);
17286: CreateAt( FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
17287: CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
17288: CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
17289: CreateFromId( Instance:THandle;ResId:Integer;ResType:PChar)
17290: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
17291: CREATENEW( AOWNER:TCOMPONENT; Dummy: Integer)
17292: CreateRes( Ident : Integer);
17293: CreateRes( MciErrNo : MCIERROR; Ident : Integer)
17294: CreateRes( ResStringRec : PResStringRec);
17295: CreateResHelp( Ident : Integer; AHelpContext : Integer);

```

```

17296: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
17297: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
17298: CreateSize( AWidth, AHeight : Integer)
17299: Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
17300:
17301: -----
17302: unit uPSI_MathMax;
17303: -----
17304: CONSTANTS
17305: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
17306: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
17307: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
17308: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
17309: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
17310: CbrtPi: Float = 1.4645918875615232630201425272638; // CubeRoot(Pi)
17311: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant
17312: PiJ: Float = 3.1415926535897932384626433832795; // PI
17313: PI: Extended = 3.1415926535897932384626433832795;
17314: PiOn2: Float = 1.5707963267948966192313216916398; // PI / 2
17315: PiOn3: Float = 1.0471975511965977461542144610932; // PI / 3
17316: PiOn4: Float = 0.78539816339744830961566084581988; // PI / 4
17317: Sqrt2: Float = 1.4142135623730950488016887242097; // Sqrt(2)
17318: Sqrt3: Float = 1.7320508075688772935274463415059; // Sqrt(3)
17319: Sqrt5: Float = 2.2360679774997896964091736687313; // Sqrt(5)
17320: Sqrt10: Float = 3.1622776601683793319988935444327; // Sqrt(10)
17321: SqrtPi: Float = 1.7724538509055160272981674833411; // Sqrt(Pi)
17322: Sqrt2Pi: Float = 2.506628274631000502415765284811; // Sqrt(2 * PI)
17323: TwoPi: Float = 6.283185307179586476925286766559; // 2 * PI
17324: ThreePi: Float = 9.424779607693797153879301498385; // 3 * PI
17325: Ln2: Float = 0.69314718055994530941723212145818; // Ln(2)
17326: Ln10: Float = 2.3025850929940456840179914546844; // Ln(10)
17327: LnPi: Float = 1.1447298858494001741434273513531; // Ln(Pi)
17328: Log2J: Float = 0.30102999566398119521373889472449; // Log10(2)
17329: Log3: Float = 0.47712125471966243729502790325512; // Log10(3)
17330: LogPi: Float = 0.4971498726941338543512682882909; // Log10(Pi)
17331: LogE: Float = 0.43429448190325182765112891891661; // Log10(E)
17332: E: Float = 2.7182818284590452353602874713527; // Natural constant
17333: hLn2Pi: Float = 0.91893853320467274178032973640562; // Ln(2*PI)/2
17334: inv2Pi: Float = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
17335: TwoToPower63: Float = 9223372036854775808.0; // 2^63
17336: GoldenMean: Float = 1.618033988749894848204586834365638; // GoldenMean
17337: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
17338: RadCor : Double = 57.29577951308232; {number of degrees in a radian}
17339: StDelta : Extended = 0.00001; {delta for difference equations}
17340: StEpsilon : Extended = 0.00001; {epsilon for difference equations}
17341: StMaxIterations : Integer = 100; {max attempts for convergence}
17342:
17343: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
17344: begin
17345: MetersPerInch = 0.0254; // [1]
17346: MetersPerFoot = MetersPerInch * 12;
17347: MetersPerYard = MetersPerFoot * 3;
17348: MetersPerMile = MetersPerFoot * 5280;
17349: MetersPerNauticalMiles = 1852;
17350: MetersPerAstronomicalUnit = 1.49598E11; // [4]
17351: MetersPerLightSecond = 2.99792458E8; // [5]
17352: MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
17353: MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
17354: MetersPerCubit = 0.4572; // [6][7]
17355: MetersPerFathom = MetersPerFoot * 6;
17356: MetersPerFurlong = MetersPerYard * 220;
17357: MetersPerHand = MetersPerInch * 4;
17358: MetersPerPace = MetersPerInch * 30;
17359: MetersPerRod = MetersPerFoot * 16.5;
17360: MetersPerChain = MetersPerRod * 4;
17361: MetersPerLink = MetersPerChain / 100;
17362: MetersPerPoint = MetersPerInch * 0.013837; // [7]
17363: MetersPerPica = MetersPerPoint * 12;
17364:
17365: SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
17366: SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
17367: SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
17368: SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
17369: SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
17370: SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
17371:
17372: CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
17373: CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
17374: CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
17375: CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
17376: CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
17377: CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
17378: CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
17379: CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
17380:
17381: CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
17382: CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
17383: CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
17384: CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;

```

```

17385: CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
17386: CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
17387: CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
17388: CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
17389:
17390: CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
17391: CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
17392: CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
17393: CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
17394: CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
17395: CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
17396:
17397: CubicMetersPerUKGallon = 0.00454609; // [2][7]
17398: CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
17399: CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
17400: CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
17401: CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
17402: CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
17403: CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
17404: CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
17405: CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
17406:
17407: GramsPerPound = 453.59237; // [1][7]
17408: GramsPerDrams = GramsPerPound / 256;
17409: GramsPerGrains = GramsPerPound / 7000;
17410: GramsPerTons = GramsPerPound * 2000;
17411: GramsPerLongTons = GramsPerPound * 2240;
17412: GramsPerOunces = GramsPerPound / 16;
17413: GramsPerStones = GramsPerPound * 14;
17414:
17415: MaxAngle 9223372036854775808.0;
17416: MaxTanH 5678.2617031470719747459655389854);
17417: MaxFactorial( 1754);
17418: MaxFloatingPoint(1.189731495357231765085759326628E+4932);
17419: MinFloatingPoint', (3.3621031431120935062626778173218E-4932);
17420: MaxTanH( 354.89135644669199842162284618659);
17421: MaxFactorial'LongInt'( 170);
17422: MaxFloatingPointD(1.797693134862315907729305190789E+308);
17423: MinFloatingPointD(2.2250738585072013830902327173324E-308);
17424: MaxTanH( 44.361419555836499802702855773323);
17425: MaxFactorial'LongInt'( 33);
17426: MaxFloatingPointS( 3.4028236692093846346337460743177E+38);
17427: MinFloatingPointS( 1.1754943508222875079687365372222E-38);
17428: PiExt( 3.1415926535897932384626433832795);
17429: RatioDegToRad( PiExt / 180.0);
17430: RatioGradToRad( PiExt / 200.0);
17431: RatioDegToGrad( 200.0 / 180.0);
17432: RatioGradToDeg( 180.0 / 200.0);
17433: Crc16PolynomCCITT'LongWord $1021);
17434: Crc16PolynomIBM'LongWord $8005);
17435: Crc16Bits'LongInt'( 16);
17436: Crc16Bytes'LongInt'( 2);
17437: Crc16HighBit'LongWord $8000);
17438: NotCrc16HighBit', 'LongWord $7FFF);
17439: Crc32PolynomIEEE', 'LongWord $04C11DB7);
17440: Crc32PolynomCastagnoli', 'LongWord $1EDC6F41);
17441: Crc32Koopman', 'LongWord $741B8CD7);
17442: Crc32Bits', 'LongInt'( 32);
17443: Crc32Bytes', 'LongInt'( 4);
17444: Crc32HighBit', 'LongWord $80000000);
17445: NotCrc32HighBit', 'LongWord $7FFFFFFF);
17446:
17447: MinByte = Low(Byte);
17448: MaxByte = High(Byte);
17449: MinWord = Low(Word);
17450: MaxWord = High(Word);
17451: MinShortInt = Low(ShortInt);
17452: MaxShortInt = High(ShortInt);
17453: MinSmallInt = Low(SmallInt);
17454: MaxSmallInt = High(SmallInt);
17455: MinLongWord = LongWord(Low(LongWord));
17456: MaxLongWord = LongWord(High(LongWord));
17457: MinLongInt = LongInt(Low(LongInt));
17458: MaxLongInt = LongInt(High(LongInt));
17459: MinInt64 = Int64(Low(Int64));
17460: MaxInt64 = Int64(High(Int64));
17461: MinInteger = Integer(Low(Integer));
17462: MaxInteger = Integer(High(Integer));
17463: MinCardinal = Cardinal(Low(Cardinal));
17464: MaxCardinal = Cardinal(High(Cardinal));
17465: MinNativeUInt = NativeUInt(Low(NativeUInt));
17466: MaxNativeUInt = NativeUInt(High(NativeUInt));
17467: MinNativeInt = NativeInt(Low(NativeInt));
17468: MaxNativeInt = NativeInt(High(NativeInt));
17469: Function CosH( const Z : Float) : Float;
17470: Function SinH( const Z : Float) : Float;
17471: Function TanH( const Z : Float) : Float;
17472:
17473:

```

```

17474: //*****from DMath.Dll Lib of types.inc in source\dmath_dll
17475: InvLn2 = 1.44269504088896340736; { 1/Ln(2) }
17476: InvLn10 = 0.43429448190325182765; { 1/Ln(10) }
17477: TwoPi = 6.28318530717958647693; { 2*Pi }
17478: PiDiv2 = 1.57079632679489661923; { Pi/2 }
17479: SqrtPi = 1.77245385090551602730; { Sqrt(Pi) }
17480: Sqrt2Pi = 2.50662827463100050242; { Sqrt(2*Pi) }
17481: InvSqrt2Pi = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
17482: LnSqrt2Pi = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
17483: Ln2PiDiv2 = 0.91893853320467274178; { Ln(2*Pi)/2 }
17484: Sqrt2 = 1.41421356237309504880; { Sqrt(2) }
17485: Sqrt2Div2 = 0.70710678118654752440; { Sqrt(2)/2 }
17486: Gold = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
17487: CGold = 0.38196601125010515179; { 2 - GOLD }
17488: MachEp = 2.220446049250313E-16; { 2^(-52) }
17489: MaxNum = 1.797693134862315E+308; { 2^1024 }
17490: MinNum = 2.225073858507202E-308; { 2^(-1022) }
17491: MaxLog = 709.7827128933840;
17492: MinLog = -708.3964185322641;
17493: MaxFac = 170;
17494: MaxGam = 171.624376956302;
17495: MaxLgm = 2.556348E+305;
17496: SingleCompareDelta = 1.0E-34;
17497: DoubleCompareDelta = 1.0E-280;
17498: { $IFDEF CLR }
17499: ExtendedCompareDelta = DoubleCompareDelta;
17500: { $ELSE }
17501: ExtendedCompareDelta = 1.0E-4400;
17502: { $ENDIF }
17503: Bytes1KB = 1024;
17504: Bytes1MB = 1024 * Bytes1KB;
17505: Bytes1GB = 1024 * Bytes1MB;
17506: Bytes64KB = 64 * Bytes1KB;
17507: Bytes64MB = 64 * Bytes1MB;
17508: Bytes2GB = 2 * LongWord(Bytes1GB);
17509: clBlack32', $FF000000 );
17510: clDimGray32', $FF3F3F3F );
17511: clGray32', $FF7F7F7F );
17512: clLightGray32', $FFBFBFBF );
17513: clWhite32', $FFFFFFF );
17514: clMaroon32', $FF7F0000 );
17515: clGreen32', $FF007F00 );
17516: clOlive32', $FF7F7F00 );
17517: clNavy32', $FF00007F );
17518: clPurple32', $FF7F007F );
17519: clTeal32', $FF007F7F );
17520: clRed32', $FFFF0000 );
17521: clLime32', $FF00FF00 );
17522: clYellow32', $FFFFFF00 );
17523: clBlue32', $FF0000FF );
17524: clFuchsia32', $FFFF00FF );
17525: clAqua32', $FF00FFFF );
17526: clAliceBlue32', $FFF0F8FF );
17527: clAntiqueWhite32', $FFFAEBD7 );
17528: clAquamarine32', $FFF7FFD4 );
17529: clAzure32', $FFF0FFFF );
17530: clBeige32', $FFF5F5DC );
17531: clBisque32', $FFFFE4C4 );
17532: clBlancheDalmont32', $FFFEBDCD );
17533: clBlueViolet32', $FF8A2BE2 );
17534: clBrown32', $FFA52A2A );
17535: clBurlyWood32', $FFDEB887 );
17536: clCadetblue32', $FF5F9EA0 );
17537: clChartReuse32', $FFF7FF00 );
17538: clChocolate32', $FFD2691E );
17539: clCoral32', $FFF7F500 );
17540: clCornFlowerBlue32', $FF6495ED );
17541: clCornSilk32', $FFFFFF8DC );
17542: clCrimson32', $FFDC143C );
17543: clDarkBlue32', $FF00008B );
17544: clDarkCyan32', $FF008B8B );
17545: clDarkGoldenRod32', $FFB8860B );
17546: clDarkGray32', $FFA9A9A9 );
17547: clDarkGreen32', $FF006400 );
17548: clDarkGrey32', $FFA9A9A9 );
17549: clDarkKhaki32', $FFBDB76B );
17550: clDarkMagenta32', $FF8B008B );
17551: clDarkOliveGreen32', $FF556B2F );
17552: clDarkOrange32', $FFF8C000 );
17553: clDarkOrchid32', $FF9932CC );
17554: clDarkRed32', $FF8B0000 );
17555: clDarkSalmon32', $FFE9967A );
17556: clDarkSeaGreen32', $FF8FBC8F );
17557: clDarkSlateBlue32', $FF483D8B );
17558: clDarkSlateGray32', $FF2F4F4F );
17559: clDarkSlateGrey32', $FF2F4F4F );
17560: clDarkTurquoise32', $FF00CED1 );
17561: clDarkViolet32', $FF9400D3 );
17562: clDeepPink32', $FFFF1493 );

```



```
17563:    clDeepSkyBlue32', $FF00BFFF );
17564:    clDodgerBlue32', $FF1E90FF );
17565:    clFireBrick32', $FFB22222 );
17566:    clFloralWhite32', $FFFFFFA0 );
17567:    clGainsBoro32', $FFDCDCDC );
17568:    clGhostWhite32', $FFF8F8FF );
17569:    clGold32', $FFFFD700 );
17570:    clGoldenRod32', $FFDAA520 );
17571:    clGreenYellow32', $FFADFF2F );
17572:    clGrey32', $FF808080 );
17573:    clHoneyDew32', $FFF0FFF0 );
17574:    clHotPink32', $FFF669B4 );
17575:    clIndianRed32', $FFCD5C5C );
17576:    clIndigo32', $FF4B0082 );
17577:    clIvory32', $FFFFFFF0 );
17578:    clKhaki32', $FFF0E68C );
17579:    clLavender32', $FFE6E6FA );
17580:    clLavenderBlush32', $FFFFFF0F5 );
17581:    clLawnGreen32', $FF7CFC00 );
17582:    clLemonChiffon32', $FFFFFFACD );
17583:    clLightBlue32', $FFADD8E6 );
17584:    clLightCoral32', $FFF08080 );
17585:    clLightCyan32', $FF00FFFF );
17586:    clLightGoldenRodYellow32', $FFFAFAD2 );
17587:    clLightGreen32', $FF90EE90 );
17588:    clLightGrey32', $FFD3D3D3 );
17589:    clLightPink32', $FFFFB6C1 );
17590:    clLightSalmon32', $FFFA07A );
17591:    clLightSeagreen32', $FF20B2AA );
17592:    clLightSkyblue32', $FF87CEFA );
17593:    clLightSlategray32', $FF778899 );
17594:    clLightSlategrey32', $FF778899 );
17595:    clLightSteelblue32', $FFB0C4DE );
17596:    clLightYellow32', $FFFFFFE0 );
17597:    clLtGray32', $FFC0C0C0 );
17598:    clMedGray32', $FFA0A0A4 );
17599:    clDkGray32', $FF808080 );
17600:    clMoneyGreen32', $FFC0DCC0 );
17601:    clLegacySkyBlue32', $FFA6CAF0 );
17602:    clCream32', $FFFFF0 );
17603:    clLimeGreen32', $FF32CD32 );
17604:    clLinen32', $FFFAF0E6 );
17605:    clMediumAquaMarine32', $FF66CDAA );
17606:    clMediumBlue32', $FF0000CD );
17607:    clMediumOrchid32', $FFBA55D3 );
17608:    clMediumPurple32', $FF9370DB );
17609:    clMediumSeaGreen32', $FF3CB371 );
17610:    clMediumSlateBlue32', $FF7B68EE );
17611:    clMediumSpringGreen32', $FF00FA9A );
17612:    clMediumTurquoise32', $FF48D1CC );
17613:    clMediumVioletRed32', $FFC71585 );
17614:    clMidnightBlue32', $FF191970 );
17615:    clMintCream32', $FFF5FFFA );
17616:    clMistyRose32', $FFFFE4E1 );
17617:    clMoccasin32', $FFFFE4B5 );
17618:    clNavajoWhite32', $FFFFDEAD );
17619:    clOldLace32', $FFFD5E6 );
17620:    clOliveDrab32', $FF6B8E23 );
17621:    clOrange32', $FFFA500 );
17622:    clOrangeRed32', $FFFA4500 );
17623:    clOrchid32', $FFDA70D6 );
17624:    clPaleGoldenRod32', $FFEE8AA );
17625:    clPaleGreen32', $FF98FB98 );
17626:    clPaleTurquoise32', $FFAFEEEE );
17627:    clPaleVioletred32', $FFDB7093 );
17628:    clPapayaWhip32', $FFFFEFD5 );
17629:    clPeachPuff32', $FFFFDAB9 );
17630:    clPeru32', $FFCD853F );
17631:    clPlum32', $FFDDA0DD );
17632:    clPowderBlue32', $FFB0E0E6 );
17633:    clRosyBrown32', $FFBC8F8F );
17634:    clRoyalBlue32', $FF4169E1 );
17635:    clSaddleBrown32', $FF8B4513 );
17636:    clSalmon32', $FFFA8072 );
17637:    clSandyBrown32', $FFFA4A60 );
17638:    clSeaGreen32', $FF2E8B57 );
17639:    clSeaShell32', $FFFFFF5EE );
17640:    clSienna32', $FFA0522D );
17641:    clSilver32', $FFC0C0C0 );
17642:    clSkyblue32', $FF87CEEB );
17643:    clSlateBlue32', $FF6A5ACD );
17644:    clSlateGray32', $FF708090 );
17645:    clSlateGrey32', $FF708090 );
17646:    clSnow32', $FFFFFFAFA );
17647:    clSpringgreen32', $FF00FF7F );
17648:    clSteelblue32', $FF4682B4 );
17649:    clTan32', $FFD2B48C );
17650:    clThistle32', $FFD8BFD8 );
17651:    clTomato32', $FFFF6347 );
```

```

17652:   clTurquoise32', $FF40E0D0 ));
17653:   clViolet32', $FFEE82EE ));
17654:   clWheat32', $FFF5DEB3 ));
17655:   clWhitesmoke32', $FFF5F5F5 ));
17656:   clYellowgreen32', $FF9ACD32 ));
17657:   clTrWhite32', $FFFFFFF ));
17658:   clTrBlack32', $7F000000 ));
17659:   clTrRed32', $7FFF0000 ));
17660:   clTrGreen32', $7F00FF00 ));
17661:   clTrBlue32', $7F0000FF ));
17662:   // Fixed point math constants
17663:   FixedOne = $10000; FixedHalf = $7FFF;
17664:   FixedPI = Round(PI * FixedOne);
17665:   FixedToFloat = 1/FixedOne;
17666:
17667:   Special Types
17668:   *****
17669:   type Complex = record
17670:     X, Y : Float;
17671:   end;
17672:   type TVector = array of Float;
17673:   TIntVector = array of Integer;
17674:   TCompVector = array of Complex;
17675:   TBoolVector = array of Boolean;
17676:   TStrVector = array of String;
17677:   TMatrix = array of TVector;
17678:   TIntMatrix = array of TIntVector;
17679:   TCompMatrix = array of TCompVector;
17680:   TBoolMatrix = array of TBoolVector;
17681:   TStrMatrix = array of TStrVector;
17682:   TByteArray = array[0..32767] of byte; !
17683:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
17684:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
17685:   T2StringArray = array of array of string;
17686:   T2IntegerArray = array of array of integer;
17687:   AddTypeS('INT_PTR', 'Integer
17688:   AddTypeS('LONG_PTR', 'Integer
17689:   AddTypeS('UINT_PTR', 'Cardinal
17690:   AddTypeS('ULONG_PTR', 'Cardinal
17691:   AddTypeS('DWORD_PTR', 'ULONG_PTR
17692:   TIntegerDynArray', 'array of Integer
17693:   TCardinalDynArray', 'array of Cardinal
17694:   TWordDynArray', 'array of Word
17695:   TSmallIntDynArray', 'array of SmallInt
17696:   TByteDynArray', 'array of Byte
17697:   TShortIntDynArray', 'array of ShortInt
17698:   TInt64DynArray', 'array of Int64
17699:   TLongWordDynArray', 'array of LongWord
17700:   TSingleDynArray', 'array of Single
17701:   TDoubleDynArray', 'array of Double
17702:   TBooleanDynArray', 'array of Boolean
17703:   TStringDynArray', 'array of string
17704:   TWideStringDynArray', 'array of WideString
17705:   TDynByteArray = array of Byte;
17706:   TDynShortintArray = array of Shortint;
17707:   TDynSmallintArray = array of Smallint;
17708:   TDynWordArray = array of Word;
17709:   TDynIntegerArray = array of Integer;
17710:   TDynLongintArray = array of Longint;
17711:   TDynCardinalArray = array of Cardinal;
17712:   TDynInt64Array = array of Int64;
17713:   TDynExtendedArray = array of Extended;
17714:   TDynDoubleArray = array of Double;
17715:   TDynSingleArray = array of Single;
17716:   TDynFloatArray = array of Float;
17717:   TDynPointerArray = array of Pointer;
17718:   TDynStringArray = array of string;
17719:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
17720:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
17721:   TSynSearchOptions = set of TSynSearchOption;
17722:
17723:
17724:
17725:   /* Project : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
17726:   -----
17727:   procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
17728:   procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
17729:   procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
17730:   procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
17731:   function CheckStringSum(vstring: string): integer;
17732:   function HexToInt(HexNum: string): LongInt;
17733:   function IntToBin(Int: Integer): String;
17734:   function BinToInt(Binary: String): Integer;
17735:   function HexToBin(HexNum: string): string; external2
17736:   function BinToHex(Binary: String): string;
17737:   function IntToFloat(i: Integer): double;
17738:   function AddThousandSeparator(S: string; myChr: Char): string;
17739:   function Max3(const X,Y,Z: Integer): Integer;
17740:   procedure Swap(var X,Y: char); // faster without inline

```

```

17741: procedure ReverseString(var S: String);
17742: function CharToHexStr(Value: Char): string;
17743: function CharToUnicode(Value: Char): string;
17744: function Hex2Dec(Value: Str002): Byte;
17745: function HexStrCodeToStr(Value: string): string;
17746: function HexToStr(i: integer; value: string): string;
17747: function UnicodeToStr(Value: string): string;
17748: function CRC16(statement: string): string;
17749: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
17750: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
17751: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
17752: procedure ExecuteCommand(executeFile, paramString: string);
17753: procedure ShellExecuteAndWait(executeFile, paramString: string);
17754: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
17755: procedure SearchAndOpenDoc(vfilenamepath: string);
17756: procedure ShowInterfaces(myFile: string);
17757: function Fact2(av: integer): extended;
17758: function BoolToStr(B: Boolean): string;
17759: function GCD(x, y: LongInt): LongInt;
17760: function LCM(m,n: longint): longint;
17761: function GetASCII: string;
17762: function GetItemHeight(Font: TFont): Integer;
17763: function myPlaySound(s: pchar; flag, syncflag: integer): boolean;
17764: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
17765: function getHINSTANCE: longword;
17766: function getHMODULE: longword;
17767: function GetASCII: string;
17768: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
17769: function WordIsOk(const AWord: string; var VW: Word): boolean;
17770: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
17771: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
17772: function SafeStr(const s: string): string;
17773: function ExtractUrlPath(const FileName: string): string;
17774: function ExtractUrlName(const FileName: string): string;
17775: function IsInternet: boolean;
17776: function RotateLeft1Bit_u32( Value: uint32): uint32;
17777: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
LF:TStLinEst; ErrorStats : Boolean);
17778: procedure getEnvironmentInfo;
17779: procedure AntiFreeze;
17780: function GetCPUSpeed: Double;
17781: function IsVirtualPcGuest : Boolean;
17782: function IsVmWareGuest : Boolean;
17783: procedure StartSerialDialog;
17784: function IsWow64: boolean;
17785: function IsWow64String(var s: string): Boolean;
17786: procedure StartThreadDemo;
17787: function RGB(R,G,B: Byte): TColor;
17788: function SendLn(amess: string): boolean;
17789: procedure maXbox;
17790: function AspectRatio(aWidth, aHeight: Integer): String;
17791: function wget(aURL, afile: string): boolean;
17792: procedure PrintList(Value: TStringList);
17793: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
17794: procedure getEnvironmentInfo;
17795: procedure AntiFreeze;
17796: function getBitmap(aphath: string): TBitmap;
17797: procedure ShowMessageBig(const aText : string);
17798: function YesNoDialog(const ACaption, AMsg: string): boolean;
17799: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
17800: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
17801: //function myStrToBytes(const Value: String): TBytes;
17802: //function myBytesToStr(const Value: TBytes): String;
17803: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
17804: function getBitmap(aphath: string): TBitmap;
17805: procedure ShowMessageBig(const aText : string);
17806: function StrToBytes(const Value: String): TBytes;
17807: function BytesToStr(const Value: TBytes): String;
17808: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
17809: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retries:Int;var
HostName:String):Bool;
17810: function FindInPaths(const fileName, paths : String) : String;
17811: procedure initHexArray(var hexn: THexArray);
17812: function josephusG(n,k: integer; var graphout: string): integer;
17813: function isPowerof2(num: int64): boolean;
17814: function powerOf2(exponent: integer): int64;
17815: function getBigPI: string;
17816: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
17817: function GetASCIILine: string;
17818: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
17819: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
17821: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
17822: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
17823: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
17824: function isKeyPressed: boolean;
17825: function KeyPress: boolean;
17826: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
17827: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;

```

```

17828: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
17829: function GetOSName: string;
17830: function GetOSVersion: string;
17831: function GetOSNumber: string;
17832: function getEnvironmentString: string;
17833: procedure StrReplace(var Str: String; Old, New: String);
17834: procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
17835: function getTeamViewerID: string;
17836: procedure RecurseDirectory(Dir: String; IncludeSubs: boolean; callback: TFileCallbackProcedure);
17837: procedure RecurseDirectory2(Dir: String; IncludeSubs: boolean);
17838: procedure WinInet_HttpGet(const Url: string; Stream: TStream);
17839: procedure GetQRCode2(Width, Height: Word; Correct_Level: string; const Data: string; apath: string);
17840: function StartSocketService: Boolean;
17841: procedure StartSocketServiceForm;
17842: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
17843: function GetFileList1(apath: string): TStringlist;
17844: procedure LetFileList(FileList: TStringlist; apath: string);
17845: procedure StartWeb(aurl: string);
17846: function GetTodayFiles(startDir, amask: string): TStringlist;
17847: function PortTCPIPisOpen(dwPort: Word; ipAddressStr: String): boolean;
17848: function JavahashCode(val: string): Integer;
17849: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
17850: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
17851: procedure HideWindowForSeconds(secs: integer); { //3 seconds }
17852: procedure HideWindowForSeconds2(secs: integer; apphandle, aSelf: TForm); { //3 seconds }
17853: procedure ConvertToGray(Cnv: TCanvas);
17854: function GetFileDate(aFile: string; aWithTime: Boolean): string;
17855: procedure ShowMemory;
17856: function ShowMemory2: string;
17857: function getHostIP: string;
17858: procedure ShowBitmap(bmap: TBitmap);
17859: function GetOsVersionInfo: TOSVersionInfo; { //thx to wischnewski }
17860: function CreateDBGridForm(dblist: TStringlist): TListbox;
17861:
17862:
17863: // News of 3.9.8 up
17864: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
17865: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
17866: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
17867: JvChart - TjvChart Component - 2009 Public
17868: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
17869: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
17870: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
17871: DMath DLL included incl. Demos
17872: Interface Navigator menu/View/Intf Navigator
17873: Unit Explorer menu/Debug/Units Explorer
17874: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
17875: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
17876: Script History to 9 Files WebServer light /Options/Addons/WebServer
17877: Full Text Finder, JVSIMLogic Simulator Package
17878: Halt-Stop Program in Menu, WebServer2, Stop Event ,
17879: Conversion Routines, Prebuild Forms, CodeSearch
17880: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
17881: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
17882: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
17883: JvChart - TjvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TjvPaintFX
17884: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
17885: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
17886: IDE Reflection API, Session Service Shell S3
17887: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
17888: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
17889: arduino map() function, PMRandom Generator
17890: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
17891: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
17892: REST Test Lib, Multilang Component, Forth Interpreter
17893: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
17894: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
17895: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
17896: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
17897: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
17898: QRCode Service, add more CFunctions like CDateTime of Synapse
17899: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
17900: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
17901: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
17902: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
17903: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
17904: BOLD Package, Indy Package5, maTRIX. MATHEMAX
17905: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
17906: emax layers: system-package-component-unit-class-function-block
17907: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms
17908: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
17909: Tutorial 18.3 RGB LED, OpenGL Geometry, maxpix, statictext
17910: OpenGL Game Demo: ..Options/Add Ons/Reversi
17911: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
17912: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
17913: 7% performance gain (hot spot profiling)
17914: PEP -Pascal Education Program, GSM Module, CGI, PHP Runner
17915: add 42 + 22 (64 units), memcached database, autobookmark, Alcino PAC, IPC Lib
17916: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units

```



```

17917: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
17918:
17919: add routines in 3.9.7.5
17920: 097: procedure RIRegister_BarCodeScanner_Routines(S: TPSExec);
17921: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
17922: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
17923: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
17924: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
17925: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
17926: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
17927:
17928: ////////////////////////////////// TestUnits //////////////////////////////////
17929: SelftestPEM;
17930: SelfTestCFundamentUtils;
17931: SelfTestCFileUtils;
17932: SelfTestCDateTime;
17933: SelfTestCTimer;
17934: SelfTestCRandom;
17935: Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
17936: Assert(WinPathToUnixPath('c\d.f') = '/c/d.f', 'WinPathToUnixPath
17937:
17938: // Note: There's no need for installing a client certificate in the
17939: // webbrowser. The server asks the webbrowser to send a certificate but
17940: // if nothing is installed the software will work because the server
17941: // doesn't check to see if a client certificate was supplied. If you want you can install:
17942: //
17943: // file: c_cacert.p12
17944: // password: c_cakey
17945:
17946: TGraphicControl = class(TControl)
17947: private
17948: FCanvas: TCanvas;
17949: procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
17950: protected
17951: procedure Paint; virtual;
17952: property Canvas: TCanvas read FCanvas;
17953: public
17954: constructor Create(AOwner: TComponent); override;
17955: destructor Destroy; override;
17956: end;
17957:
17958: TCustomControl = class(TWinControl)
17959: private
17960: FCanvas: TCanvas;
17961: procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
17962: protected
17963: procedure Paint; virtual;
17964: procedure PaintWindow(DC: HDC); override;
17965: property Canvas: TCanvas read FCanvas;
17966: public
17967: constructor Create(AOwner: TComponent); override;
17968: destructor Destroy; override;
17969: end;
17970: RegisterPublishedProperties;
17971: ('ONCHANGE', 'TNotifyEvent', iptrw);
17972: ('ONCLICK', 'TNotifyEvent', iptrw);
17973: ('ONDBLCLICK', 'TNotifyEvent', iptrw);
17974: ('ONENTER', 'TNotifyEvent', iptrw);
17975: ('ONEXIT', 'TNotifyEvent', iptrw);
17976: ('ONKEYDOWN', 'TKeyEvent', iptrw);
17977: ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
17978: ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
17979: ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
17980: ('ONMOUSEUP', 'TMouseEvent', iptrw);
17981: //////////////////////////////////////////
17982: // To stop the while loop, click on Options/Show Include (boolean switch)!
17983: Control a loop in a script with a form event:
17984: IncludeON; //control the while loop
17985: while maxform1.ShowInclude1.checked do begin //menu event Options/Show Include
17986:
17987: //-----
17988: //*****mX4 ini-file Configuration*****
17989: //-----
17990: using config file maxboxdef.ini menu/Help/Config File
17991:
17992: **** Definitions for maxbox mX3 ***
17993: [FORM]
17994: LAST_FILE=E:\maxbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
17995: FONTSIZE=14
17996: EXTENSION=txt
17997: SCREENX=1386
17998: SCREENY=1077
17999: MEMHEIGHT=350
18000: PRINTFONT=Courier New //GUI Settings
18001: LINENUMBERS=Y //line numbers at gutter in editor at left side
18002: EXCEPTIONLOG=Y //store excepts and success in 2 log files see below! - menu Debug/Show Last Exceptions
18003: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
18004: BOOTSCRIPT=Y //enabling load a boot script
18005: MEMORYREPORT=Y //shows memory report on closing maxbox

```

```

18006: MACRO=Y           //expand macros (see below) in code e.g. #path:E:\maxbox\maxbox3\docs\
18007: NAVIGATOR=N       //shows function list at the right side of editor
18008: NAVWIDTH=350      //width of the right side interface list <CTRL L>
18009: AUTOBOOKMARK=Y   //sets on all functions a bookmark to jump
18010: [WEB]
18011: IPPORT=8080        //for internal webserver - menu /Options/Add Ons/WebServer
18012: IPHOST=192.168.1.53
18013: ROOTCERT=filepathY
18014: SCERT=filepathY
18015: RSAKEY=filepathY
18016: VERSIONCHECK=Y
18017:
18018: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
18019:
18020: Also possible to set report memory in script to override ini setting
18021: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
18022:
18023:
18024: After Change the ini file you can reload the file with ../Help/Config Update
18025:
18026: //-----
18027: //*****mX4 maildef.ini ini-file Configuration*****
18028: //-----
18029: /*** Definitions for maxMail ***/
18030: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
18031: [MAXMAIL]
18032: HOST=getmail.softwareschule.ch
18033: USER=mailusername
18034: PASS=password
18035: PORT=110
18036: SSL=Y
18037: BODY=Y
18038: LAST=5
18039:
18040: ADO Connection String:
18041: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
18042:
18043: OpenSSL Lib: unit ssl_openssl_lib;
18044: {$IFDEF CIL}
18045: const
18046:   {$IFDEF LINUX}
18047:     DLLSSLName = 'libssl.so';
18048:     DLLUtilName = 'libcrypto.so';
18049:   {$ELSE}
18050:     DLLSSLName = 'ssleay32.dll';
18051:     DLLUtilName = 'libeay32.dll';
18052:   {$ENDIF}
18053: {$ELSE}
18054: var
18055:   {$IFDEF MSWINDOWS}
18056:     {$IFDEF DARWIN}
18057:       DLLSSLName: string = 'libssl.dylib';
18058:       DLLUtilName: string = 'libcrypto.dylib';
18059:     {$ELSE}
18060:       DLLSSLName: string = 'libssl.so';
18061:       DLLUtilName: string = 'libcrypto.so';
18062:     {$ENDIF}
18063:   {$ELSE}
18064:     DLLSSLName: string = 'ssleay32.dll';
18065:     DLLSSLName2: string = 'libssl32.dll';
18066:     DLLUtilName: string = 'libeay32.dll';
18067:   {$ENDIF}
18068: {$ENDIF}
18069:
18070:
18071:
18072: //-----
18073: //*****mX4 Macro Tags *****
18074: //-----
18075:
18076: asm #name #hostmAPSN2APSN2111e, #head,max: APSN21: 04.01.2014 19:05:50
E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
18077:
18078: //Tag Macros
18079:
18080: asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
18081:
18082: //Tag Macros
18083: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
18084: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
18085: 10190: SearchAndCopy(memo1.lines, '#host', getComputernamewin, 11);
18086: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
18087: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
18088: 10199: SearchAndCopy(memo1.lines, '#fils', fname + ' +SHA1(Act_Filename), 11);
18089: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
18090: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
18091: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
18092:   [getUserNameWin, getComputernamewin, datetimetoStr(now)],
18093: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',

```

```

18094: 10197: [getUserNameWin, getComputernamewin, datetimetToStr(now), Act_Filename]],11);
18095: [getUserNameWin, getComputernamewin, datetimetToStr(now), Act_Filename]],11);
18096: 10198: SearchAndCopy(memol.lines, '#tech',Format('perf: %s threads: %d %s %s',
18097: [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]], 11);
18098: 10298: SearchAndCopy(memol.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
18099: [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)], 10);
18100:
18101: // #tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
18102:
18103: //Replace Macros
18104: SearchAndCopy(memol.lines, '<TIME>', timetoStr(time), 6);
18105: SearchAndCopy(memol.lines, '<DATE>', datetoStr(date), 6);
18106: SearchAndCopy(memol.lines, '<PATH>', fpath, 6);
18107: SearchAndCopy(memol.lines, '<EXEPATH>', EXEPATH, 9);
18108: SearchAndCopy(memol.lines, '<FILE>', fname, 6);
18109: SearchAndCopy(memol.lines, '<SOURCE>', ExePath+'Source', 8);
18110:
18111: 10198: SearchAndCopy(memol.lines, '#tech!perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
18112: [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]], 11);
18113: // #tech!84perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
18114: SearchAndCopy(memol.lines, 'maxbox_extract_funclist399.txt
18115:
18116: //-----
18117: //*****mX4 ToDo List Tags ../Help/ToDo List*****
18118: //-----
18119:
18120: while I < sl.Count do begin
18121: // if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:') then
18122: if MatchesMask(sl[I], '*/? TODO (?*#?#)*:') then
18123: BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
18124: else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:') then
18125: BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
18126: else if MatchesMask(sl[I], '*/? TODO (?*#?#)*:') then
18127: BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
18128: else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:') then
18129: BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
18130: else if MatchesMask(sl[I], '*/?*TODO*:') then
18131: BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
18132: else if MatchesMask(sl[I], '*/?*DONE*:') then
18133: BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
18134: Inc(I);
18135: end;
18136:
18137:
18138: //-----
18139: //*****mX4 Public Tools API *****
18140: //-----
18141: file : unit uPSI_fMain.pas; {SOTAP} Open Tools API Catalog
18142: // Those functions concern the editor and preprocessor, all of the IDE
18143: Example: Call it with maxform1.InfolClick(self)
18144: Note: Call all Methods with maxForm1., e.g.:
18145: maxForm1.ShellStyle1Click(self);
18146:
18147: procedure SIRegister_fMain(CL: TPSPascalCompiler);
18148: begin
18149: Const('BYTECODE','String 'bytecode.txt'
18150: Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
18151: Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
18152: Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
18153: Const('PSINC','String PS Includes (*.inc)|*.INC
18154: Const('DEFFILENAME','String 'firstdemo.txt
18155: Const('DEFINIFILE','String 'maxboxdef.ini
18156: Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
18157: Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
18158: Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
18159: Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
18160: Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
18161: Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
18162: Const('INCLUDEBOX','String 'pas_includebox.inc
18163: Const('BOOTSCRIPT','String 'maxbootscript.txt
18164: Const('MBVERSION','String '3.9.9.94
18165: Const('VERSION','String '3.9.9.94
18166: Const('MBVER','String '399
18167: Const('MBVERI','Integer'(399);
18168: Const('MBVERIALI','Integer'(39994);
18169: Const('EXENAME','String 'maXbox3.exe
18170: Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
18171: Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
18172: Const('MXINTERNETCHECK','String 'www.ask.com
18173: Const('MXMAIL','String 'max@kleiner.com
18174: Const('TAB','Char #S09);
18175: Const('CODECOMPLETION','String 'bds_delphi.dci
18176: SIRegister_TMaxForm1(CL);
18177: end;
18178:
18179: with FindClass('TForm'),'TMaxForm1' do begin
18180: memo2', 'TMemo', iptrw);
18181: memo1', 'TSynMemo', iptrw);
18182: CB1SCList', 'TComboBox', iptrw);

```

```
18183: mxNavigator', 'TComboBox', iptrw);
18184: IPhost', 'string', iptrw);
18185: IPPort', 'integer', iptrw);
18186: COMPort', 'integer', iptrw); //3.9.6.4
18187: Splitter1', 'TSplitter', iptrw);
18188: PSScript', 'TPSScript', iptrw);
18189: PS3DllPlugin', 'TPSDllPlugin', iptrw);
18190: MainMenu1', 'TMainMenu', iptrw);
18191: Program1', 'TMenuItem', iptrw);
18192: Compile1', 'TMenuItem', iptrw);
18193: Files1', 'TMenuItem', iptrw);
18194: open1', 'TMenuItem', iptrw);
18195: Save2', 'TMenuItem', iptrw);
18196: Options1', 'TMenuItem', iptrw);
18197: Savebefore1', 'TMenuItem', iptrw);
18198: Largefont1', 'TMenuItem', iptrw);
18199: sBytecodel', 'TMenuItem', iptrw);
18200: Saveas3', 'TMenuItem', iptrw);
18201: Clear1', 'TMenuItem', iptrw);
18202: Slinenumbers1', 'TMenuItem', iptrw);
18203: About1', 'TMenuItem', iptrw);
18204: Search1', 'TMenuItem', iptrw);
18205: SynPasSyn1', 'TSynPasSyn', iptrw);
18206: memol', 'TSynMemo', iptrw);
18207: SynEditSearch1', 'TSynEditSearch', iptrw);
18208: WordWrap1', 'TMenuItem', iptrw);
18209: XPManifest1', 'TXPManifest', iptrw);
18210: SearchNext1', 'TMenuItem', iptrw);
18211: Replacel', 'TMenuItem', iptrw);
18212: PSImport_Controls1', 'TPSImport_Controls', iptrw);
18213: PSImport_Classes1', 'TPSImport_Classes', iptrw);
18214: ShowIncludel', 'TMenuItem', iptrw);
18215: SynEditPrint1', 'TSynEditPrint', iptrw);
18216: Printout1', 'TMenuItem', iptrw);
18217: mnPrintColors1', 'TMenuItem', iptrw);
18218: dlgFilePrint', 'TPrintDialog', iptrw);
18219: dlgPrintFont1', 'TFontDialog', iptrw);
18220: mnuPrintFont1', 'TMenuItem', iptrw);
18221: Includel', 'TMenuItem', iptrw);
18222: CodeCompletionList1', 'TMenuItem', iptrw);
18223: IncludeList1', 'TMenuItem', iptrw);
18224: ImageList1', 'TImageList', iptrw);
18225: ImageList2', 'TImageList', iptrw);
18226: CoolBar1', 'TCoolBar', iptrw);
18227: ToolBar1', 'TToolBar', iptrw);
18228: tbtnLoad', 'TToolButton', iptrw);
18229: ToolButton2', 'TToolButton', iptrw);
18230: tbtnFind', 'TToolButton', iptrw);
18231: tbtnCompile', 'TToolButton', iptrw);
18232: tbtnTrans', 'TToolButton', iptrw);
18233: tbtnUseCase', 'TToolButton', iptrw); //3.8
18234: toolbtnTutorial', 'TToolButton', iptrw);
18235: tbtn6res', 'TToolButton', iptrw);
18236: ToolButton5', 'TToolButton', iptrw);
18237: ToolButton1', 'TToolButton', iptrw);
18238: ToolButton3', 'TToolButton', iptrw);
18239: statusBar1', 'TStatusBar', iptrw);
18240: SaveOutput1', 'TMenuItem', iptrw);
18241: ExportClipboard1', 'TMenuItem', iptrw);
18242: Closel', 'TMenuItem', iptrw);
18243: Manual1', 'TMenuItem', iptrw);
18244: About2', 'TMenuItem', iptrw);
18245: loadLastfile1', 'TMenuItem', iptrw);
18246: imglogo', 'TImage', iptrw);
18247: cedebug', 'TPSScriptDebugger', iptrw);
18248: debugPopupMenu1', 'TPopupMenu', iptrw);
18249: BreakPointMenu', 'TMenuItem', iptrw);
18250: Decompile1', 'TMenuItem', iptrw);
18251: StepInto1', 'TMenuItem', iptrw);
18252: StepOut1', 'TMenuItem', iptrw);
18253: Reset1', 'TMenuItem', iptrw);
18254: DebugRun1', 'TMenuItem', iptrw);
18255: PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
18256: PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
18257: PSImport_Forms1', 'TPSImport_Forms', iptrw);
18258: PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
18259: tutorial4', 'TMenuItem', iptrw);
18260: ExporttoClipboard1', 'TMenuItem', iptrw);
18261: ImportfromClipboard1', 'TMenuItem', iptrw);
18262: N4', 'TMenuItem', iptrw);
18263: N5', 'TMenuItem', iptrw);
18264: N6', 'TMenuItem', iptrw);
18265: ImportfromClipboard2', 'TMenuItem', iptrw);
18266: tutorial1', 'TMenuItem', iptrw);
18267: N7', 'TMenuItem', iptrw);
18268: ShowSpecChars1', 'TMenuItem', iptrw);
18269: OpenDirectory1', 'TMenuItem', iptrw);
18270: procMess', 'TMenuItem', iptrw);
18271: tbtnUseCase', 'TToolButton', iptrw);
```



```
18272: ToolButton7', 'TToolButton', iptrw);
18273: EditFont1', 'TMenuItem', iptrw);
18274: UseCase1', 'TMenuItem', iptrw);
18275: tutorial21', 'TMenuItem', iptrw);
18276: OpenUseCase1', 'TMenuItem', iptrw);
18277: PSImport_DB1', 'TPSImport_DB', iptrw);
18278: tutorial31', 'TMenuItem', iptrw);
18279: SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
18280: HTMLSyntax1', 'TMenuItem', iptrw);
18281: ShowInterfaces1', 'TMenuItem', iptrw);
18282: Tutorial5', 'TMenuItem', iptrw);
18283: AllFunctionsList1', 'TMenuItem', iptrw);
18284: ShowLastException1', 'TMenuItem', iptrw);
18285: PlayMP31', 'TMenuItem', iptrw);
18286: SynTeXSyn1', 'TSynTeXSyn', iptrw);
18287: texSyntax1', 'TMenuItem', iptrw);
18288: N8', 'TMenuItem', iptrw);
18289: GetEmails1', 'TMenuItem', iptrw);
18290: SynCppSyn1', 'TSynCppSyn', iptrw);
18291: CSyntax1', 'TMenuItem', iptrw);
18292: Tutorial6', 'TMenuItem', iptrw);
18293: New1', 'TMenuItem', iptrw);
18294: AllObjectsList1', 'TMenuItem', iptrw);
18295: LoadBytecode1', 'TMenuItem', iptrw);
18296: CipherFile1', 'TMenuItem', iptrw);
18297: N9', 'TMenuItem', iptrw);
18298: N10', 'TMenuItem', iptrw);
18299: Tutorial11', 'TMenuItem', iptrw);
18300: Tutorial71', 'TMenuItem', iptrw);
18301: UpdateService1', 'TMenuItem', iptrw);
18302: PascalSchool1', 'TMenuItem', iptrw);
18303: Tutorial81', 'TMenuItem', iptrw);
18304: DelphiSite1', 'TMenuItem', iptrw);
18305: Output1', 'TMenuItem', iptrw);
18306: TerminalStyle1', 'TMenuItem', iptrw);
18307: ReadOnly1', 'TMenuItem', iptrw);
18308: ShellStyle1', 'TMenuItem', iptrw);
18309: BigScreen1', 'TMenuItem', iptrw);
18310: Tutorial91', 'TMenuItem', iptrw);
18311: SaveOutput2', 'TMenuItem', iptrw);
18312: N11', 'TMenuItem', iptrw);
18313: SaveScreenshot', 'TMenuItem', iptrw);
18314: Tutorial101', 'TMenuItem', iptrw);
18315: SQLSyntax1', 'TMenuItem', iptrw);
18316: SynSQLSyn1', 'TSynSQLSyn', iptrw);
18317: Console1', 'TMenuItem', iptrw);
18318: SynXMLSyn1', 'TSynXMLSyn', iptrw);
18319: XMLSyntax1', 'TMenuItem', iptrw);
18320: ComponentCount1', 'TMenuItem', iptrw);
18321: NewInstance1', 'TMenuItem', iptrw);
18322: toolbtnTutorial', 'TToolButton', iptrw);
18323: Memory1', 'TMenuItem', iptrw);
18324: SynJavaSyn1', 'TSynJavaSyn', iptrw);
18325: JavaSyntax1', 'TMenuItem', iptrw);
18326: SyntaxCheck1', 'TMenuItem', iptrw);
18327: Tutorial10Statistics1', 'TMenuItem', iptrw);
18328: ScriptExplorer1', 'TMenuItem', iptrw);
18329: FormOutput1', 'TMenuItem', iptrw);
18330: ArduinoDump1', 'TMenuItem', iptrw);
18331: AndroidDump1', 'TMenuItem', iptrw);
18332: GotoEnd1', 'TMenuItem', iptrw);
18333: AllResourceList1', 'TMenuItem', iptrw);
18334: ToolButton4', 'TToolButton', iptrw);
18335: tbtn6res', 'TToolButton', iptrw);
18336: Tutorial11Forms1', 'TMenuItem', iptrw);
18337: Tutorial12SQL1', 'TMenuItem', iptrw);
18338: ResourceExplore1', 'TMenuItem', iptrw);
18339: Info1', 'TMenuItem', iptrw);
18340: N12', 'TMenuItem', iptrw);
18341: CryptoBox1', 'TMenuItem', iptrw);
18342: Tutorial13Ciphering1', 'TMenuItem', iptrw);
18343: CipherFile2', 'TMenuItem', iptrw);
18344: N13', 'TMenuItem', iptrw);
18345: ModulesCount1', 'TMenuItem', iptrw);
18346: AddOns2', 'TMenuItem', iptrw);
18347: N4GewinntGame1', 'TMenuItem', iptrw);
18348: DocuforAddOns1', 'TMenuItem', iptrw);
18349: Tutorial14Async1', 'TMenuItem', iptrw);
18350: Lessons15Review1', 'TMenuItem', iptrw);
18351: SynPHPSyn1', 'TSynPHPSyn', iptrw);
18352: PHPSyntax1', 'TMenuItem', iptrw);
18353: Breakpoint1', 'TMenuItem', iptrw);
18354: SerialRS2321', 'TMenuItem', iptrw);
18355: N14', 'TMenuItem', iptrw);
18356: SynCSSSyn1', 'TSynCSSSyn', iptrw);
18357: CSyntax2', 'TMenuItem', iptrw);
18358: Calculator1', 'TMenuItem', iptrw);
18359: tbtnSerial', 'TToolButton', iptrw);
18360: ToolButton8', 'TToolButton', iptrw);
```

```
18361: Tutorial151', 'TMenuItem', iptrw);
18362: N15', 'TMenuItem', iptrw);
18363: N16', 'TMenuItem', iptrw);
18364: ControlBar1', 'TControlBar', iptrw);
18365: ToolBar2', 'TToolBar', iptrw);
18366: BtnOpen', 'TToolButton', iptrw);
18367: BtnSave', 'TToolButton', iptrw);
18368: BtnPrint', 'TToolButton', iptrw);
18369: BtnColors', 'TToolButton', iptrw);
18370: btnClassReport', 'TToolButton', iptrw);
18371: BtnRotateRight', 'TToolButton', iptrw);
18372: BtnFullSize', 'TToolButton', iptrw);
18373: BtnFitToWindowSize', 'TToolButton', iptrw);
18374: BtnZoomMinus', 'TToolButton', iptrw);
18375: BtnZoomPlus', 'TToolButton', iptrw);
18376: Panell1', 'TPanel', iptrw);
18377: LabelBrettgroesse', 'TLabel', iptrw);
18378: CB1SCList', 'TComboBox', iptrw);
18379: ImageListNormal', 'TImageList', iptrw);
18380: spbtnexplore', 'TSpeedButton', iptrw);
18381: spbtnexample', 'TSpeedButton', iptrw);
18382: spbsaveas', 'TSpeedButton', iptrw);
18383: imglogobox', 'TImage', iptrw);
18384:EnlargeFont1', 'TMenuItem', iptrw);
18385:EnlargeFont2', 'TMenuItem', iptrw);
18386:ShrinkFont1', 'TMenuItem', iptrw);
18387:ThreadDemol', 'TMenuItem', iptrw);
18388:HEXEditor1', 'TMenuItem', iptrw);
18389:HEXView1', 'TMenuItem', iptrw);
18390:HEXInspect1', 'TMenuItem', iptrw);
18391:SynExporterHTML1', 'TSynExporterHTML', iptrw);
18392:ExporttoHTML1', 'TMenuItem', iptrw);
18393:ClassCount1', 'TMenuItem', iptrw);
18394:HTMLOutput1', 'TMenuItem', iptrw);
18395:HEXEditor2', 'TMenuItem', iptrw);
18396:Minesweeper1', 'TMenuItem', iptrw);
18397: N17', 'TMenuItem', iptrw);
18398: PicturePuzzle1', 'TMenuItem', iptrw);
18399: sbvclhelp', 'TSpeedButton', iptrw);
18400: DependencyWalker1', 'TMenuItem', iptrw);
18401: WebScanner1', 'TMenuItem', iptrw);
18402: View1', 'TMenuItem', iptrw);
18403: mnToolbar1', 'TMenuItem', iptrw);
18404: mnStatusbar2', 'TMenuItem', iptrw);
18405: mnConsole2', 'TMenuItem', iptrw);
18406: mnCoolbar2', 'TMenuItem', iptrw);
18407: mnSplitter2', 'TMenuItem', iptrw);
18408: WebServer1', 'TMenuItem', iptrw);
18409: Tutorial17Server1', 'TMenuItem', iptrw);
18410: Tutorial18Arduinol', 'TMenuItem', iptrw);
18411: SynPerlSyn1', 'TSynPerlSyn', iptrw);
18412: PerlSyntax1', 'TMenuItem', iptrw);
18413: SynPythonSyn1', 'TSynPythonSyn', iptrw);
18414: PythonSyntax1', 'TMenuItem', iptrw);
18415: DMathLibrary1', 'TMenuItem', iptrw);
18416: IntfNavigator1', 'TMenuItem', iptrw);
18417:EnlargeFontConsole1', 'TMenuItem', iptrw);
18418:ShrinkFontConsole1', 'TMenuItem', iptrw);
18419:SetInterfaceList1', 'TMenuItem', iptrw);
18420: popintfList', 'TPopupMenu', iptrw);
18421: intfAdd1', 'TMenuItem', iptrw);
18422: intfDelete1', 'TMenuItem', iptrw);
18423: intfRefactor1', 'TMenuItem', iptrw);
18424: Defactor1', 'TMenuItem', iptrw);
18425: Tutorial19COMArduinol', 'TMenuItem', iptrw);
18426: Tutorial20Regex', 'TMenuItem', iptrw);
18427: N18', 'TMenuItem', iptrw);
18428: ManualE1', 'TMenuItem', iptrw);
18429: FullTextFinder1', 'TMenuItem', iptrw);
18430: Move1', 'TMenuItem', iptrw);
18431: FractalDemol', 'TMenuItem', iptrw);
18432: Tutorial21Android1', 'TMenuItem', iptrw);
18433: Tutorial0Function1', 'TMenuItem', iptrw);
18434: SimuLogBox1', 'TMenuItem', iptrw);
18435: OpenExamples1', 'TMenuItem', iptrw);
18436: SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
18437: JavaScriptSyntax1', 'TMenuItem', iptrw);
18438: Halt1', 'TMenuItem', iptrw);
18439: CodeSearch1', 'TMenuItem', iptrw);
18440: SynRubySyn1', 'TSynRubySyn', iptrw);
18441: RubySyntax1', 'TMenuItem', iptrw);
18442: Undo1', 'TMenuItem', iptrw);
18443: SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
18444: LinuxShellScript1', 'TMenuItem', iptrw);
18445: Rename1', 'TMenuItem', iptrw);
18446: spdcodesearch', 'TSpeedButton', iptrw);
18447: Preview1', 'TMenuItem', iptrw);
18448: Tutorial22Services1', 'TMenuItem', iptrw);
18449: Tutorial23RealTime1', 'TMenuItem', iptrw);
```

```

18450: Configuration1', 'TMenuItem', iptrw);
18451: MP3Player1', 'TMenuItem', iptrw);
18452: DLLSpy1', 'TMenuItem', iptrw);
18453: SynURIOpener1', 'TSynURIOpener', iptrw);
18454: SynURISyn1', 'TSynURISyn', iptrw);
18455: URILinksClicks1', 'TMenuItem', iptrw);
18456: EditReplacer1', 'TMenuItem', iptrw);
18457: GotoLine1', 'TMenuItem', iptrw);
18458: ActiveLineColor1', 'TMenuItem', iptrw);
18459: ConfigFile1', 'TMenuItem', iptrw);
18460: SortIntfList1', 'TMenuItem', iptrw);
18461: Redol1', 'TMenuItem', iptrw);
18462: Tutorial24CleanCode1', 'TMenuItem', iptrw);
18463: Tutorial25Configuration1', 'TMenuItem', iptrw);
18464: IndentSelection1', 'TMenuItem', iptrw);
18465: UnindentSection1', 'TMenuItem', iptrw);
18466: SkyStyle1', 'TMenuItem', iptrw);
18467: N19', 'TMenuItem', iptrw);
18468: CountWords1', 'TMenuItem', iptrw);
18469: imbookmarkimages', 'TImageList', iptrw);
18470: Bookmark11', 'TMenuItem', iptrw);
18471: N20', 'TMenuItem', iptrw);
18472: Bookmark21', 'TMenuItem', iptrw);
18473: Bookmark31', 'TMenuItem', iptrw);
18474: Bookmark41', 'TMenuItem', iptrw);
18475: SynMultiSyn1', 'TSynMultiSyn', iptrw);
18476:
18477: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
18478: Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
18479: Procedure PSScriptCompile( Sender : TPSScript)
18480: Procedure Compile1Click( Sender : TObject)
18481: Procedure PSScriptExecute( Sender : TPSScript)
18482: Procedure open1Click( Sender : TObject)
18483: Procedure Save2Click( Sender : TObject)
18484: Procedure Savebefore1Click( Sender : TObject)
18485: Procedure Largefont1Click( Sender : TObject)
18486: Procedure FormActivate( Sender : TObject)
18487: Procedure SBytecode1Click( Sender : TObject)
18488: Procedure FormKeyPress( Sender : TObject; var Key : Char)
18489: Procedure Saveas3Click( Sender : TObject)
18490: Procedure Clear1Click( Sender : TObject)
18491: Procedure Slinenumbers1Click( Sender : TObject)
18492: Procedure About1Click( Sender : TObject)
18493: Procedure Search1Click( Sender : TObject)
18494: Procedure FormCreate( Sender : TObject)
18495: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
18496:                               var Action : TSynReplaceAction)
18497: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
18498: Procedure WordWrap1Click( Sender : TObject)
18499: Procedure SearchNext1Click( Sender : TObject)
18500: Procedure Replacer1Click( Sender : TObject)
18501: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
18502: Procedure ShowInclude1Click( Sender : TObject)
18503: Procedure Printout1Click( Sender : TObject)
18504: Procedure mnuPrintFont1Click( Sender : TObject)
18505: Procedure Include1Click( Sender : TObject)
18506: Procedure FormDestroy( Sender : TObject)
18507: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
18508: Procedure UpdateView1Click( Sender : TObject)
18509: Procedure CodeCompletionList1Click( Sender : TObject)
18510: Procedure SaveOutput1Click( Sender : TObject)
18511: Procedure ExportClipboard1Click( Sender : TObject)
18512: Procedure Close1Click( Sender : TObject)
18513: Procedure Manual1Click( Sender : TObject)
18514: Procedure LoadLastFile1Click( Sender : TObject)
18515: Procedure Memo1Change( Sender : TObject)
18516: Procedure Decompile1Click( Sender : TObject)
18517: Procedure StepInto1Click( Sender : TObject)
18518: Procedure StepOut1Click( Sender : TObject)
18519: Procedure Reset1Click( Sender : TObject)
18520: Procedure cedebugAfterExecute( Sender : TPSScript)
18521: Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
18522: Procedure cedebugCompile( Sender : TPSScript)
18523: Procedure cedebugExecute( Sender : TPSScript)
18524: Procedure cedebugIdle( Sender : TObject)
18525: Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
18526: Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
18527: Procedure BreakPointMenuClick( Sender : TObject)
18528: Procedure DebugRun1Click( Sender : TObject)
18529: Procedure tutorial4Click( Sender : TObject)
18530: Procedure ImportfromClipboard1Click( Sender : TObject)
18531: Procedure ImportfromClipboard2Click( Sender : TObject)
18532: Procedure tutorial1Click( Sender : TObject)
18533: Procedure ShowSpecChars1Click( Sender : TObject)
18534: Procedure StatusBar1Db1Click( Sender : TObject)
18535: Procedure PSScriptLine( Sender : TObject)
18536: Procedure OpenDirectory1Click( Sender : TObject)
18537: Procedure procMessClick( Sender : TObject)
18538: Procedure tbtnUseCaseClick( Sender : TObject)

```

```

18539: Procedure EditFont1Click( Sender : TObject)
18540: Procedure tutorial21Click( Sender : TObject)
18541: Procedure tutorial31Click( Sender : TObject)
18542: Procedure HTMLSyntax1Click( Sender : TObject)
18543: Procedure ShowInterfaces1Click( Sender : TObject)
18544: Procedure Tutorial5Click( Sender : TObject)
18545: Procedure ShowLastException1Click( Sender : TObject)
18546: Procedure PlayMP31Click( Sender : TObject)
18547: Procedure AllFunctionsList1Click( Sender : TObject)
18548: Procedure texSyntax1Click( Sender : TObject)
18549: Procedure GetEMails1Click( Sender : TObject)
18550: procedure DelphiSite1Click(Sender: TObject);
18551: procedure TerminalStyle1Click(Sender: TObject);
18552: procedure ReadOnly1Click(Sender: TObject);
18553: procedure ShellStyle1Click(Sender: TObject);
18554: procedure Console1Click(Sender: TObject); //3.2
18555: procedure BigScreen1Click(Sender: TObject);
18556: procedure Tutorial91Click(Sender: TObject);
18557: procedure SaveScreenshotClick(Sender: TObject);
18558: procedure Tutorial101Click(Sender: TObject);
18559: procedure SQLSyntax1Click(Sender: TObject);
18560: procedure XMLSyntax1Click(Sender: TObject);
18561: procedure ComponentCount1Click(Sender: TObject);
18562: procedure NewInstance1Click(Sender: TObject);
18563: procedure CSyntax1Click(Sender: TObject);
18564: procedure Tutorial6Click(Sender: TObject);
18565: procedure New1Click(Sender: TObject);
18566: procedure AllObjectsList1Click(Sender: TObject);
18567: procedure LoadBytecode1Click(Sender: TObject);
18568: procedure CipherFile1Click(Sender: TObject); //V3.5
18569: procedure NewInstance1Click(Sender: TObject);
18570: procedure toolbtnTutorialClick(Sender: TObject);
18571: procedure Memory1Click(Sender: TObject);
18572: procedure JavaSyntax1Click(Sender: TObject);
18573: procedure SyntaxCheck1Click(Sender: TObject);
18574: procedure ScriptExplorer1Click(Sender: TObject);
18575: procedure FormOutput1Click(Sender: TObject); //V3.6
18576: procedure GotoEnd1Click(Sender: TObject);
18577: procedure AllResourceList1Click(Sender: TObject);
18578: procedure tbtn6resClick(Sender: TObject); //V3.7
18579: procedure Info1Click(Sender: TObject);
18580: procedure Tutorial10Statistics1Click(Sender: TObject);
18581: procedure Tutorial11Forms1Click(Sender: TObject);
18582: procedure Tutorial12SQL1Click(Sender: TObject); //V3.8
18583: procedure ResourceExplore1Click(Sender: TObject);
18584: procedure Info1Click(Sender: TObject);
18585: procedure CryptoBox1Click(Sender: TObject);
18586: procedure ModulesCount1Click(Sender: TObject);
18587: procedure N4GewinntGame1Click(Sender: TObject);
18588: procedure PHPSyntax1Click(Sender: TObject);
18589: procedure SerialRS2321Click(Sender: TObject);
18590: procedure CSyntax2Click(Sender: TObject);
18591: procedure Calculator1Click(Sender: TObject);
18592: procedure Tutorial13Ciphering1Click(Sender: TObject);
18593: procedure Tutorial14Async1Click(Sender: TObject);
18594: procedure PHPSyntax1Click(Sender: TObject);
18595: procedure BtnZoomPlusClick(Sender: TObject);
18596: procedure BtnZoomMinusClick(Sender: TObject);
18597: procedure btnClassReportClick(Sender: TObject);
18598: procedure ThreadDemo1Click(Sender: TObject);
18599: procedure HEXView1Click(Sender: TObject);
18600: procedure ExporttoHTML1Click(Sender: TObject);
18601: procedure Minesweeper1Click(Sender: TObject);
18602: procedure PicturePuzzle1Click(Sender: TObject); //V3.9
18603: procedure sbvclhelpClick(Sender: TObject);
18604: procedure DependencyWalker1Click(Sender: TObject);
18605: procedure CB1SCListDrawItem(Control: TWinControl; Index: Int; aRect: TRect; State: TOwnerDrawState);
18606: procedure WebScanner1Click(Sender: TObject);
18607: procedure mnToolbar1Click(Sender: TObject);
18608: procedure mnStatusBar2Click(Sender: TObject);
18609: procedure mnConsole2Click(Sender: TObject);
18610: procedure mnCoolbar2Click(Sender: TObject);
18611: procedure mnSplitter2Click(Sender: TObject);
18612: procedure WebServer1Click(Sender: TObject);
18613: procedure PerlSyntax1Click(Sender: TObject);
18614: procedure PythonSyntax1Click(Sender: TObject);
18615: procedure DMathLibrary1Click(Sender: TObject);
18616: procedure IntfNavigator1Click(Sender: TObject);
18617: procedure FullTextFinder1Click(Sender: TObject);
18618: function AppName: string;
18619: function ScriptName: string;
18620: function LastName: string;
18621: procedure FractalDemo1Click(Sender: TObject);
18622: procedure SimuLogBox1Click(Sender: TObject);
18623: procedure OpenExamples1Click(Sender: TObject);
18624: procedure Halt1Click(Sender: TObject);
18625: procedure Stop;
18626: procedure CodeSearch1Click(Sender: TObject);
18627: procedure RubySyntax1Click(Sender: TObject);

```



```

18628: procedure Undo1Click(Sender: TObject);
18629: procedure LinuxShellScript1Click(Sender: TObject);
18630: procedure WebScannerDirect(urls: string);
18631: procedure WebScanner(urls: string);
18632: procedure LoadInterfaceList2;
18633: procedure DLLSpy1Click(Sender: TObject);
18634: procedure Memo1Db1Click(Sender: TObject);
18635: procedure URILinksClicks1Click(Sender: TObject);
18636: procedure GotoLine1Click(Sender: TObject);
18637: procedure ConfigFile1Click(Sender: TObject);
18638: procedure Sort1IntfListClick( Sender : TObject)
18639: Procedure Redo1Click( Sender : TObject)
18640: Procedure Tutorial24CleanCode1Click( Sender : TObject)
18641: Procedure IndentSection1Click( Sender : TObject)
18642: Procedure UnindentSection1Click( Sender : TObject)
18643: Procedure SkyStyle1Click( Sender : TObject)
18644: Procedure CountWords1Click( Sender : TObject)
18645: Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
18646: Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
18647: Procedure Bookmark11Click( Sender : TObject)
18648: Procedure Bookmark21Click( Sender : TObject)
18649: Procedure Bookmark31Click( Sender : TObject)
18650: Procedure Bookmark41Click( Sender : TObject)
18651: Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operation:TRangeOperation;var Range:Pointer);
18652: 'STATMemoryReport', 'boolean', iptrw);
18653: 'IPPort', 'integer', iptrw);
18654: 'COMPort', 'integer', iptrw);
18655: 'lbintfList', 'TListBox', iptrw);
18656: Function GetStatChange : boolean
18657: Procedure SetStatChange( vstat : boolean)
18658: Function GetActFileName : string
18659: Procedure SetActFileName( vname : string)
18660: Function GetLastFileName : string
18661: Procedure SetLastFileName( vname : string)
18662: Procedure WebScannerDirect( urls : string)
18663: Procedure LoadInterfaceList2
18664: Function GetStatExecuteShell : boolean
18665: Procedure DoEditorExecuteCommand( EditorCommand : word)
18666: function GetActiveLineColor: TColor
18667: procedure SetActiveLineColor(acolor: TColor)
18668: procedure ScriptListBox1Click(Sender: TObject);
18669: procedure Memo2KeyPress(Sender: TObject; var Key: Char);
18670: procedure EnlargeGutter1Click(Sender: TObject);
18671: procedure Tetris1Click(Sender: TObject);
18672: procedure ToDoList1Click(Sender: TObject);
18673: procedure ProcessList1Click(Sender: TObject);
18674: procedure MetricReport1Click(Sender: TObject);
18675: procedure ProcessList1Click(Sender: TObject);
18676: procedure TCPSockets1Click(Sender: TObject);
18677: procedure ConfigUpdate1Click(Sender: TObject);
18678: procedure ADOWorkbench1Click(Sender: TObject);
18679: procedure SocketServer1Click(Sender: TObject);
18680: procedure FormDemolClick(Sender: TObject);
18681: procedure Richedit1Click(Sender: TObject);
18682: procedure SimpleBrowser1Click(Sender: TObject);
18683: procedure DOSShell1Click(Sender: TObject);
18684: procedure SynExport1Click(Sender: TObject);
18685: procedure ExporttoRTF1Click(Sender: TObject);
18686: procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
18687: procedure SOAPTester1Click(Sender: TObject);
18688: procedure Sniffer1Click(Sender: TObject);
18689: procedure AutoDetectSyntax1Click(Sender: TObject);
18690: procedure FPlot1Click(Sender: TObject);
18691: procedure PasStyle1Click(Sender: TObject);
18692: procedure Tutorial118RGBLED1Click(Sender: TObject);
18693: procedure ReversilClick(Sender: TObject);
18694: procedure ManualmaXbox1Click(Sender: TObject);
18695: procedure BlaisePascalMagazine1Click(Sender: TObject);
18696: procedure AddToDo1Click(Sender: TObject);
18697: procedure CreateGUID1Click(Sender: TObject);
18698: procedure Tutorial27XML1Click(Sender: TObject);
18699: procedure CreateDLLStub1Click(Sender: TObject);
18700: procedure Tutorial28DLL1Click(Sender: TObject);
18701: procedure ResetKeyPressed;
18702: procedure FileChanges1Click(Sender: TObject);
18703: procedure OpenGLTry1Click(Sender: TObject);
18704: procedure AllUnitList1Click(Sender: TObject);
18705: procedure Tutorial29UMLClick(Sender: TObject);
18706: procedure CreateHeader1Click(Sender: TObject);
18707:
18708: //-----
18709: //*****mX4 Editor SynEdit Tools API *****
18710: //-----
18711: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
18712: begin
18713:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
18714:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
18715:     Constructor Create( AOwner : TComponent)
18716:     SelStart, 'Integer', iptrw);

```

```

18717: SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
18718: Procedure UpdateCaret
18719: Procedure AddKey(Command: TSynEditorCommand; Key1: word; SS1: TShiftState; Key2: word; SS2: TShiftState);
18720: Procedure AddKey(Command: TSynEditorCommand; Key1: word; SS1: TShiftState; Key2: word; SS2: TShiftState);
18721: Procedure BeginUndoBlock
18722: Procedure BeginUpdate
18723: Function CaretInView : Boolean
18724: Function CharIndexToRowCol( Index : integer ) : TBufferCoord
18725: Procedure Clear
18726: Procedure ClearAll
18727: Procedure ClearBookMark( BookMark : Integer)
18728: Procedure ClearSelection
18729: Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
18730: Procedure ClearUndo
18731: Procedure CopyToClipboard
18732: Procedure CutToClipboard
18733: Procedure DoCopyToClipboard( const SText : string)
18734: Procedure EndUndoBlock
18735: Procedure EndUpdate
18736: Procedure EnsureCursorPosVisible
18737: Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
18738: Procedure FindMatchingBracket
18739: Function GetMatchingBracket : TBufferCoord
18740: Function GetMatchingBracketEx( const APoint : TBufferCoord ) : TBufferCoord
18741: Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
18742: Function GetBookMark( BookMark : integer; var X, Y : integer ) : boolean
18743: Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
      : TSynHighlighterAttributes ) : boolean
18744: Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
      var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
18745: Function GetPositionOfMouse( out aPos : TBufferCoord ) : Boolean
18746: Function GetWordAtRowCol( const XY : TBufferCoord ) : string
18747: Procedure GotoBookMark( BookMark : Integer)
18748: Procedure GotoLineAndCenter( ALine : Integer)
18749: Function IdentChars : TSynIdentChars
18750: Procedure InvalidateGutter
18751: Procedure InvalidateGutterLine( aLine : integer)
18752: Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
18753: Procedure InvalidateLine( Line : integer)
18754: Procedure InvalidateLines( FirstLine, LastLine : integer)
18755: Procedure InvalidateSelection
18756: Function IsBookmark( BookMark : integer ) : boolean
18757: Function IsPointInSelection( const Value : TBufferCoord ) : boolean
18758: Procedure LockUndo
18759: Function BufferToDisplayPos( const p : TBufferCoord ) : TDisplayCoord
18760: Function DisplayToBufferPos( const p : TDisplayCoord ) : TBufferCoord
18761: Function LineToRow( aLine : integer ) : integer
18762: Function RowToLine( aRow : integer ) : integer
18763: Function NextWordPos : TBufferCoord
18764: Function NextWordPosEx( const XY : TBufferCoord ) : TBufferCoord
18765: Procedure PasteFromClipboard
18766: Function WordStart : TBufferCoord
18767: Function WordStartEx( const XY : TBufferCoord ) : TBufferCoord
18768: Function WordEnd : TBufferCoord
18769: Function WordEndEx( const XY : TBufferCoord ) : TBufferCoord
18770: Function PrevWordPos : TBufferCoord
18771: Function PrevWordPosEx( const XY : TBufferCoord ) : TBufferCoord
18772: Function PixelsToRowColumn( aX, aY : integer ) : TDisplayCoord
18773: Function PixelsToNearestRowColumn( aX, aY : integer ) : TDisplayCoord
18774: Procedure Redo
18775: Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer));
18776: Function RowColumnToPixels( const RowCol : TDisplayCoord ) : TPoint
18777: Function RowColToCharIndex( RowCol : TBufferCoord ) : integer
18778: Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
18779: Procedure SelectAll
18780: Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
18781: Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
18782: Procedure SetDefaultKeystrokes
18783: Procedure SetSelWord
18784: Procedure SetWordBlock( Value : TBufferCoord)
18785: Procedure Undo
18786: Procedure UnlockUndo
18787: Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
18788: Procedure AddKeyUpHandler( aHandler : TKeyEvent)
18789: Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
18790: Procedure AddKeyDownHandler( aHandler : TKeyEvent)
18791: Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
18792: Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
18793: Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
18794: Procedure AddFocusControl( aControl : TWinControl)
18795: Procedure RemoveFocusControl( aControl : TWinControl)
18796: Procedure AddMouseDownHandler( aHandler : TMouseEvent)
18797: Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
18798: Procedure AddMouseUpHandler( aHandler : TMouseEvent)
18799: Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
18800: Procedure AddMouseCursorHandler( aHandler : TMouseEvent)
18801: Procedure RemoveMouseCursorHandler( aHandler : TMouseEvent)
18802: Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
18803: Procedure RemoveLinesPointer
18804:
18805:

```

```

18806:   Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
18807:   Procedure UnHookTextBuffer
18808:   BlockBegin', 'TBufferCoord', iptrw);
18809:   BlockEnd', 'TBufferCoord', iptrw);
18810:   CanPaste', 'Boolean', iptr);
18811:   CanRedo', 'boolean', iptr);
18812:   CanUndo', 'boolean', iptr);
18813:   CaretX', 'Integer', iptrw);
18814:   CaretY', 'Integer', iptrw);
18815:   CaretXY', 'TBufferCoord', iptrw);
18816:   ActiveLineColor', 'TColor', iptrw);
18817:   DisplayX', 'Integer', iptr);
18818:   DisplayY', 'Integer', iptr);
18819:   DisplayXY', 'TDisplayCoord', iptr);
18820:   DisplayLineCount', 'integer', iptr);
18821:   CharsInWindow', 'Integer', iptr);
18822:   CharWidth', 'integer', iptr);
18823:   Font', 'TFont', iptrw);
18824:   GutterWidth', 'Integer', iptr);
18825:   Highlighter', 'TSynCustomHighlighter', iptrw);
18826:   LeftChar', 'Integer', iptrw);
18827:   LineHeight', 'integer', iptr);
18828:   LinesInWindow', 'Integer', iptr);
18829:   LineText', 'string', iptrw);
18830:   Lines', 'TStrings', iptrw);
18831:   Marks', 'TSynEditMarkList', iptr);
18832:   MaxScrollWidth', 'integer', iptrw);
18833:   Modified', 'Boolean', iptrw);
18834:   PaintLock', 'Integer', iptr);
18835:   ReadOnly', 'Boolean', iptrw);
18836:   SearchEngine', 'TSynEditSearchCustom', iptrw);
18837:   SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
18838:   SelTabBlock', 'Boolean', iptr);
18839:   SelTabLine', 'Boolean', iptr);
18840:   SelText', 'string', iptrw);
18841:   StateFlags', 'TSynStateFlags', iptr);
18842:   Text', 'string', iptrw);
18843:   TopLine', 'Integer', iptrw);
18844:   WordAtCursor', 'string', iptr);
18845:   WordAtMouse', 'string', iptr);
18846:   UndoList', 'TSynEditUndoList', iptr);
18847:   RedoList', 'TSynEditUndoList', iptr);
18848:   OnProcessCommand', 'TProcessCommandEvent', iptrw);
18849:   BookMarkOptions', 'TSynBookMarkOpt', iptrw);
18850:   BorderStyle', 'TSynBorderStyle', iptrw);
18851:   ExtraLineSpacing', 'integer', iptrw);
18852:   Gutter', 'TSynGutter', iptrw);
18853:   HideSelection', 'boolean', iptrw);
18854:   InsertCaret', 'TSynEditCaretType', iptrw);
18855:   InsertMode', 'boolean', iptrw);
18856:   IsScrolling', 'Boolean', iptr);
18857:   Keystrokes', 'TSynEditKeyStrokes', iptrw);
18858:   MaxUndo', 'Integer', iptrw);
18859:   Options', 'TSynEditorOptions', iptrw);
18860:   OverwriteCaret', 'TSynEditCaretType', iptrw);
18861:   RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
18862:   ScrollHintColor', 'TColor', iptrw);
18863:   ScrollHintFormat', 'TScrollHintFormat', iptrw);
18864:   ScrollBars', 'TScrollStyle', iptrw);
18865:   SelectedColor', 'TSynSelectedColor', iptrw);
18866:   SelectionMode', 'TSynSelectionMode', iptrw);
18867:   ActiveSelectionMode', 'TSynSelectionMode', iptrw);
18868:   TabWidth', 'integer', iptrw); WantReturns', 'boolean', iptrw);
18869:   WantTabs', 'boolean', iptrw); WordWrap', 'boolean', iptrw);
18870:   WordWrapGlyph', 'TSynGlyph', iptrw);
18871:   OnChange', 'TNotifyEvent', iptrw);
18872:   OnClearBookmark', 'TPlaceMarkEvent', iptrw);
18873:   OnCommandProcessed', 'TProcessCommandEvent', iptrw);
18874:   OnContextHelp', 'TContextHelpEvent', iptrw);
18875:   OnDropFiles', 'TDropFilesEvent', iptrw);
18876:   OnGutterClick', 'TGutterClickEvent', iptrw);
18877:   OnGutterGetText', 'TGutterGetTextEvent', iptrw);
18878:   OnGutterPaint', 'TGutterPaintEvent', iptrw);
18879:   OnMouseCursor', 'TMouseCursorEvent', iptrw);
18880:   OnPaint', 'TPaintEvent', iptrw);
18881:   OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
18882:   OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
18883:   OnReplaceText', 'TReplaceTextEvent', iptrw);
18884:   OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
18885:   OnStatusChange', 'TStatusChangeEvent', iptrw);
18886:   OnPaintTransient', 'TPaintTransient', iptrw);
18887:   OnScroll', 'TScrollEvent', iptrw);
18888: end;
18889: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
18890: Function GetPlaceableHighlighters : TSynHighlighterList
18891: Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
18892: Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
18893: Procedure GetEditorCommandValues( Proc : TGetStrProc)
18894: Procedure GetEditorCommandExtended( Proc : TGetStrProc)

```

```

18895: Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
18896: Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
18897: Function ConvertCodeStringToExtended( AString : String) : String
18898: Function ConvertExtendedToCodeString( AString : String) : String
18899: Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
18900: Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
18901: Function IndexToEditorCommand( const AIndex : Integer) : Integer
18902:
18903: TSynEditorOption = (
18904:   eoAltSetsColumnMode,           //Holding down the Alt Key will put the selection mode into columnar format
18905:   eoAutoIndent,                 //Will indent caret on newlines with same amount of leading whitespace as
18906:                                 // preceding line
18907:   eoAutoSizeMaxScrollWidth,     //Automatically resizes the MaxScrollWidth property when inserting text
18908:   eoDisableScrollArrows,        //Disables the scroll bar arrow buttons when you can't scroll in that
18909:                                 //direction any more
18910:   eoDragDropEditing,            //Allows to select a block of text and drag it within document to another
18911:                                 // location
18912:   eoDropFiles,                  //Allows the editor accept OLE file drops
18913:   eoEnhanceHomeKey,             //Enhances home key positioning, similar to visual studio
18914:   eoEnhanceEndKey,              //enhances End key positioning, similar to JDeveloper
18915:   eoGroupUndo,                  //When undoing/redoin actions, handle all continous changes the same kind
18916:                                 // in one call
18917:                                 //instead undoing/redoin each command separately
18918:   eoHalfPageScroll,             //When scrolling with page-up and page-down commands, only scroll a half
18919:                                 //page at a time
18920:   eoHideShowScrollbars,         //if enabled, then scrollbars will only show if necessary.
18921:   If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
18922:   eoKeepCaretX,                 //When moving through lines w/o cursor Past EOL, keeps X position of cursor
18923:   eoNoCaret,                     //Makes it so the caret is never visible
18924:   eoNoSelection,                 //Disables selecting text
18925:   eoRightMouseMovesCursor,      //When clicking with right mouse for popup menu, moves cursor to location
18926:   eoScrollByOneLess,             //Forces scrolling to be one less
18927:   eoScrollHintFollows,           //The scroll hint follows the mouse when scrolling vertically
18928:   eoScrollPastEof,              //Allows the cursor to go past the end of file marker
18929:   eoScrollPastEol,              //Allows cursor to go past last character into white space at end of a line
18930:   eoShowScrollHint,             //Shows a hint of the visible line numbers when scrolling vertically
18931:   eoShowSpecialChars,           //Shows the special characters
18932:   eoSmartTabDelete,             //similar to Smart Tabs, but when you delete characters
18933:   eoSmartTabs,                  //When tabbing, cursor will go to non-white space character of previous line
18934:   eoSpecialLineDefaultFg,       //disables the foreground text color override using OnSpecialLineColor event
18935:   eoTabIndent,                  //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
18936:   eoTabsToSpaces,               //Converts a tab character to a specified number of space characters
18937:   eoTrimTrailingSpaces          //Spaces at the end of lines will be trimmed and not saved
18938:
18939:   *****Important Editor Short Cuts*****);
18940: Double click to select a word and count words with highlightning.
18941: Triple click to select a line.
18942: CTRL+SHIFT+click to extend a selection.
18943: Drag with the ALT key down to select columns of text !!!
18944: Drag and drop is supported.
18945: Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
18946: Type CTRL+A to select all.
18947: Type CTRL+N to set a new line.
18948: Type CTRL+T to delete a line or token. //Tokenizer
18949: Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
18950: Type CTRL+Shift+T to add ToDo in line and list.
18951: Type CTRL+Shift+[0..9] to set bookmarks. //Bookmark
18952: Type CTRL[0..9] to jump or get to bookmarks.
18953: Type Home to position cursor at beginning of current line and End to position it at end of line.
18954: Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
18955: Page Up and Page Down work as expected.
18956: CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
18957: using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
18958:
18959: {$ Short Key Positions Ctrl<A-Z>: }
18960: def
18961:   <A> Select All
18962:   <B> Count Words
18963:   <C> Copy
18964:   <D> Internet Start
18965:   <E> Script List
18966:   <F> Find
18967:   <G> Goto
18968:   <H> Mark Line
18969:   <I> Interface List
18970:   <J> Code Completion
18971:   <K> Console
18972:   <L> Interface List Box
18973:   <M> Font Larger -
18974:   <N> New Line
18975:   <O> Open File
18976:   <P> Font Smaller +
18977:   <Q> Quit
18978:   <R> Replace
18979:   <S> Save!
18980:   <T> Delete Line
18981:   <U> Use Case Editor
18982:   <V> Paste
18983:   <W> URI Links

```



```

18984: <X> Reserved for coding use internal
18985: <Y> Delete Line
18986: <Z> Undo
18987:
18988: ref
18989: F1 Help
18990: F2 Syntax Check
18991: F3 Search Next
18992: F4 New Instance
18993: F5 Line Mark /Breakpoint
18994: F6 Goto End
18995: F7 Debug Step Into
18996: F8 Debug Step Out
18997: F9 Compile
18998: F10 Menu
18999: F11 Word Count Highlight
19000: F12 Reserved for coding use internal
19001:
19002: def ReservedWords: array[0..82] of string =
19003:   ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
19004:    'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
19005:    'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
19006:    'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
19007:    'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
19008:    'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
19009:    'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
19010:    'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
19011:    'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
19012:    'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
19013:    'public', 'published', 'def', 'ref', 'using', 'typedef', 'memol', 'memo2', 'doc', 'maxform1';
19014: AllowedChars: array[0..5] of string = (('(',')','[',']','{','}','t,t1,t2,t3: boolean;
19015:
19016: //-----
19017: //*****End of mX4 Public Tools API *****
19018: //-----
19019:
19020: Amount of Functions: 12264
19021: Amount of Procedures: 7632
19022: Amount of Constructors: 1253
19023: Totals of Calls: 21149
19024: SHA1: Win 3.9.9.94 746418954E790A1713A93FF786ABC50097FCB59
19025:
19026: *****
19027: Doc Short Manual with 50 Tips!
19028: *****
19029: - Install: just save your maxboxdef.ini before and then extract the zip file!
19030: - Toolbar: Click on the red maxbox Sign (right on top) opens your work directory or jump to <Help>
19031: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
19032: - Menu: With <Ctrl><F3> you can search for code on examples
19033: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
19034: - Menu: Set Interface Navigator in menu /View/Intf Navigator
19035: - Menu: Switch or toggle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
19036:
19037: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
19038: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
19039: - Context Menu: You can printout your scripts as a pdf-file or html-export
19040: - Context: You do have a context menu with the right mouse click
19041:
19042: - Menu: With the UseCase Editor you can convert graphic formats too.
19043: - Menu: On menu Options you find Addons as compiled scripts
19044: - IDE: You don't need a mouse to handle maxbox, use shortcuts
19045: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
19046: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
19047: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
19048:           or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
19049:
19050: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
19051: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
19052: - Code: If you code a loop till key-pressed use function: isKeyPressed;
19053: - Code: Macro set the macros #name, #date, #host, #path, #file, #head #sign, Tutorial maxbox_starter25.pdf
19054: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
19055: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
19056:           to delete and Click and mark to drag a bookmark
19057: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
19058: - IDE: A file info with system and script information you find in menu Program/Information
19059: - IDE: After change the config file in help you can update changes in menu Help/Config Update
19060: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
19061: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
19062: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
19063: - Editor: Set Bookmarks to check your work in app or code
19064: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
19065: - Editor: With {/TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
19066: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
19067:
19068: - IDE with menu /Options/ADO SQL Workbench you can manage your Database
19069: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
19070: - Menu: Set Interface Navigator also with toggle <Ctrl L> or /View/Intf Navigator
19071: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
19072: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;

```

```

19073: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
19074: - IDE set bookmarks with <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
19075: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
19076:
19077: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
19078: - Add on when no browser is available start /Options/Add ons/Easy Browser
19079: - Add on SOAP Tester with SOP POST File
19080: - Add on IP Protocol Sniffer with List View
19081: - Add on OpenGL mX Robot Demo for android
19082:
19083: - Menu: Help/Tools as a Tool Section with DOS Opener
19084: - Menu Editor: export the code as RTF File
19085: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
19086: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
19087: - Context: Auto Detect of Syntax depending on file extension
19088: - Code: some Windows API function start with w in the name like wGetAtomName();
19089: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
19090: - IDE File Check with menu ..View/File Changes/...
19091:
19092: - using DLL example in maxbox: //function: {*****}
19093:   Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
19094:     cb: DWORD): BOOL; //stdcall;
19095:   External 'GetProcessMemoryInfo@psapi.dll stdcall';
19096:   Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
19097:   External 'OpenProcess@kernel32.dll stdcall';
19098:
19099:
19100: PCT Precompile Technology , mX4 ScriptStudio
19101: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
19102: DMATH, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
19103: emax layers: system-package-component-unit-class-function-block
19104: new keywords def ref using maxCalcF
19105: UML: use case act class state seq pac comp dep - lib lab
19106: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
19107: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
19108: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
19109: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
19110: 1 DLL Report, 24 Units add, DRTTable, Remote+, Cindy functions!
19111: DLL Report, UML Tutor, 32 Units add, DRTTable, Remote+, Cindy functions!
19112: https://unibe-ch.academia.edu/MaxKleiner
19113: www.slideshare.net/maxkleiner1
19114: http://www.scribd.com/max_kleiner
19115:
19116:
19117:
19118: *****
19119:   unit List asm internal end
19120: *****
19121: 01 unit RRegister_StrUtils_Routines(exec); //Delphi
19122: 02 unit SRegister_IdStrings //Indy Sockets
19123: 03 unit RRegister_niSTRING_Routines(Exec); //from RegEx
19124: 04 unit uPSI_fMain Functions; //maxbox Open Tools API
19125: 05 unit IFSI_WinFormpuzzle; //maxbox
19126: 06 unit RRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
19127: 07 unit RegisterDateTimeLibrary_R(exec); //Delphi
19128: 08 unit RRegister_MathMax_Routines(exec); //Jedi & Delphi
19129: 09 unit RRegister_IdGlobal_Routines(exec); //Indy Sockets
19130: 10 unit RRegister_SysUtils_Routines(Exec); //Delphi
19131: 11 unit uPSI_IdTCPConnection; //Indy some functions
19132: 12 unit uPSCompiler.pas; //PS kernel functions
19133: 13 unit uPSI_DBCommon; //DB Common_Routines and Types
19134: 14 unit uPSI_Printers.pas //Delphi VCL
19135: 15 unit uPSI_MPlayer.pas //Delphi VCL
19136: 16 unit uPSC_comobj; //COM Functions
19137: 17 unit uPSI_Clipbrd; //Delphi VCL
19138: 18 unit Filectrl in IFSI_SysUtils_max; //VCL Runtime
19139: 19 unit uPSI_SqlExpr; //DBX3
19140: 20 unit uPSI_ADODB; //ADODB
19141: 21 unit uPSI_StrHlpr; //String Helper Routines
19142: 22 unit uPSI_DateUtils; //Expansion to DateTimeLib
19143: 23 unit uPSI_FileUtils; //Expansion to Sys/File Utils
19144: 24 unit JUtils / gsUtils; //Jedi / Metabase
19145: 25 unit JvFunctions_max; //Jedi Functions
19146: 26 unit HTTPParser; //Delphi VCL
19147: 27 unit HTTPUtil; //Delphi VCL
19148: 28 unit uPSI_XMLUtil; //Delphi VCL
19149: 29 unit uPSI_SOAPHTTIClient; //Delphi VCL SOAP WebService V3.5
19150: 30 unit uPSI_Contnr; //Delphi RTL Container of Classes
19151: 31 unit uPSI_MaskUtils; //RTL Edit and Mask functions
19152: 32 unit uPSI_MyBigInt; //big integer class with Math
19153: 33 unit uPSI_ConvUtils; //Delphi VCL Conversions engine
19154: 34 unit Types_Variants; //Delphi\Win32\rtl\sys
19155: 35 unit uPSI_IdHashSHA1; //Indy Crypto Lib
19156: 36 unit uPSI_IdHashMessageDigest //Indy Crypto;
19157: 37 unit uPSI_IdASNIUtil; //Indy ASN1Utility Routines;
19158: 38 unit uPSI_IdLogFile; //Indy Logger from LogBase
19159: 39 unit uPSI_IdIcmpClient; //Indy Ping ICMP
19160: 40 unit uPSI_IdHashMessageDigest_max //Indy Crypto &OpenSSL;
19161: 41 unit uPSI_FileCtrl; //Delphi RTL

```

```

19162: 42 unit uPSI_Outline; //Delphi VCL
19163: 43 unit uPSI_ScktComp; //Delphi RTL
19164: 44 unit uPSI_Calendar; //Delphi VCL
19165: 45 unit uPSI_VListVw; //VListVw;
19166: 46 unit uPSI_DBGGrids; //Delphi VCL
19167: 47 unit uPSI_DBCtrls; //Delphi VCL
19168: 48 unit ide_debugoutput; //maXbox
19169: 49 unit uPSI_ComCtrls; //Delphi VCL
19170: 50 unit uPSC_stdCtrls; //Delphi VCL
19171: 51 unit uPSI_Dialogs; //Delphi VCL
19172: 52 unit uPSI_StdConv; //Delphi RTL
19173: 53 unit uPSI_DBClient; //Delphi RTL
19174: 54 unit uPSI_DBPlatform; //Delphi RTL
19175: 55 unit uPSI_Provider; //Delphi RTL
19176: 56 unit uPSI_FMTBcd; //Delphi RTL
19177: 57 unit uPSI_DBCGrids; //Delphi VCL
19178: 58 unit uPSI_CDSUtil; //MIDAS
19179: 59 unit uPSI_VarHlpr; //Delphi RTL
19180: 60 unit uPSI_ExtDlgs; //Delphi VCL
19181: 61 unit sdpStopwatch; //maXbox
19182: 62 unit uPSI_JclStatistics; //JCL
19183: 63 unit uPSI_JclLogic; //JCL
19184: 64 unit uPSI_JclMiscel; //JCL
19185: 65 unit uPSI_JclMath_max; //JCL RTL
19186: 66 unit uPSI_uTPLb_StreamUtils; //LockBox 3
19187: 67 unit uPSI_MathUtils; //BCB
19188: 68 unit uPSI_JclMultimedia; //JCL
19189: 69 unit uPSI_WideStrUtils; //Delphi API/RTL
19190: 70 unit uPSI_GraphUtil; //Delphi RTL
19191: 71 unit uPSI_TypeTrans; //Delphi RTL
19192: 72 unit uPSI_HTTPApp; //Delphi VCL
19193: 73 unit uPSI_DBWeb; //Delphi VCL
19194: 74 unit uPSI_DBBdeWeb; //Delphi VCL
19195: 75 unit uPSI_DBXpressWeb; //Delphi VCL
19196: 76 unit uPSI_ShadowWnd; //Delphi VCL
19197: 77 unit uPSI_ToolWin; //Delphi VCL
19198: 78 unit uPSI_Tabs; //Delphi VCL
19199: 79 unit uPSI_JclGraphUtils; //JCL
19200: 80 unit uPSI_JclCounter; //JCL
19201: 81 unit uPSI_JclSysInfo; //JCL
19202: 82 unit uPSI_JclSecurity; //JCL
19203: 83 unit uPSI_JclFileUtils; //JCL
19204: 84 unit uPSI_IdUserAccounts; //Indy
19205: 85 unit uPSI_IdAuthentication; //Indy
19206: 86 unit uPSI_uTPLb_AES; //LockBox 3
19207: 87 unit uPSI_IdHashSHA1; //LockBox 3
19208: 88 unit uTPLb_BlockCipher; //LockBox 3
19209: 89 unit uPSI_ValEdit.pas; //Delphi VCL
19210: 90 unit uPSI_JvVCLUtils; //JCL
19211: 91 unit uPSI_JvDBUtil; //JCL
19212: 92 unit uPSI_JvDBUtils; //JCL
19213: 93 unit uPSI_JvAppUtils; //JCL
19214: 94 unit uPSI_JvCtrlUtils; //JCL
19215: 95 unit uPSI_JvFormToHtml; //JCL
19216: 96 unit uPSI_JvParsing; //JCL
19217: 97 unit uPSI_SerDlgs; //Toolbox
19218: 98 unit uPSI_Serial; //Toolbox
19219: 99 unit uPSI_JvComponent; //JCL
19220: 100 unit uPSI_JvCalc; //JCL
19221: 101 unit uPSI_JvBdeUtils; //JCL
19222: 102 unit uPSI_JvDateUtil; //JCL
19223: 103 unit uPSI_JvGenetic; //JCL
19224: 104 unit uPSI_JclBase; //JCL
19225: 105 unit uPSI_JvUtils; //JCL
19226: 106 unit uPSI_JvStrUtil; //JCL
19227: 107 unit uPSI_JvStrUtils; //JCL
19228: 108 unit uPSI_JvFileUtil; //JCL
19229: 109 unit uPSI_JvMemoryInfos; //JCL
19230: 110 unit uPSI_JvComputerInfo; //JCL
19231: 111 unit uPSI_JvgCommClasses; //JCL
19232: 112 unit uPSI_JvgLogics; //JCL
19233: 113 unit uPSI_JvLED; //JCL
19234: 114 unit uPSI_JvTurtle; //JCL
19235: 115 unit uPSI_SortThds; unit uPSI_ThSort; //maXbox
19236: 116 unit uPSI_JvgUtils; //JCL
19237: 117 unit uPSI_JvExprParser; //JCL
19238: 118 unit uPSI_HexDump; //Borland
19239: 119 unit uPSI_DBLogDlg; //VCL
19240: 120 unit uPSI_SqlTimSt; //RTL
19241: 121 unit uPSI_JvHtmlParser; //JCL
19242: 122 unit uPSI_JvgXMLSerializer; //JCL
19243: 123 unit uPSI_JvJCLUtils; //JCL
19244: 124 unit uPSI_JvStrings; //JCL
19245: 125 unit uPSI_uTPLb_IntegerUtils; //TurboPower
19246: 126 unit uPSI_uTPLb_HugeCardinal; //TurboPower
19247: 127 unit uPSI_uTPLb_HugeCardinalUtils; //TurboPower
19248: 128 unit uPSI_SynRegExpr; //SynEdit
19249: 129 unit uPSI_StUtils; //SysTools4
19250: 130 unit uPSI_StToHTML; //SysTools4

```

```

19251: 131 unit uPSI_StStrms; //SysTools4
19252: 132 unit uPSI_StFIN; //SysTools4
19253: 133 unit uPSI_StAstroP; //SysTools4
19254: 134 unit uPSI_StStat; //SysTools4
19255: 135 unit uPSI_StNetCon; //SysTools4
19256: 136 unit uPSI_StDecMth; //SysTools4
19257: 137 unit uPSI_StOStr; //SysTools4
19258: 138 unit uPSI_StPtrns; //SysTools4
19259: 139 unit uPSI_StNetMessage; //SysTools4
19260: 140 unit uPSI_StMath; //SysTools4
19261: 141 unit uPSI_StExpEng; //SysTools4
19262: 142 unit uPSI_StCRC; //SysTools4
19263: 143 unit uPSI_StExport; //SysTools4
19264: 144 unit uPSI_StExpLog; //SysTools4
19265: 145 unit uPSI_ActnList; //Delphi VCL
19266: 146 unit uPSI_jpeg; //Borland
19267: 147 unit uPSI_StRandom; //SysTools4
19268: 148 unit uPSI_StDict; //SysTools4
19269: 149 unit uPSI_StBCD; //SysTools4
19270: 150 unit uPSI_StTxtDat; //SysTools4
19271: 151 unit uPSI_StRegEx; //SysTools4
19272: 152 unit uPSI_IMouse; //VCL
19273: 153 unit uPSI_SyncObjs; //VCL
19274: 154 unit uPSI_AsyncCalls; //Hausladen
19275: 155 unit uPSI_ParallelJobs; //Saraiva
19276: 156 unit uPSI_Variants; //VCL
19277: 157 unit uPSI_VarCmplx; //VCL Wolfram
19278: 158 unit uPSI_DTDSchema; //VCL
19279: 159 unit uPSI_ShLwApi; //Brakel
19280: 160 unit uPSI_IBUtils; //VCL
19281: 161 unit uPSI_CheckLst; //VCL
19282: 162 unit uPSI_JvSimpleXml; //JCL
19283: 163 unit uPSI_JclSimpleXml; //JCL
19284: 164 unit uPSI_JvXmlDatabase; //JCL
19285: 165 unit uPSI_JvMaxPixel; //JCL
19286: 166 unit uPSI_JvItemsSearchs; //JCL
19287: 167 unit uPSI_StExpEng2; //SysTools4
19288: 168 unit uPSI_StGenLog; //SysTools4
19289: 169 unit uPSI_JvLogFile; //Jcl
19290: 170 unit uPSI_CPort; //ComPort Lib v4.11
19291: 171 unit uPSI_CPortCtl; //ComPort
19292: 172 unit uPSI_CPortEsc; //ComPort
19293: 173 unit BarCodeScanner; //ComPort
19294: 174 unit uPSI_JvGraph; //JCL
19295: 175 unit uPSI_JvComCtrls; //JCL
19296: 176 unit uPSI_GUITesting; //D Unit
19297: 177 unit uPSI_JvFindFiles; //JCL
19298: 178 unit uPSI_StSystem; //SysTools4
19299: 179 unit uPSI_JvKeyboardStates; //JCL
19300: 180 unit uPSI_JvMail; //JCL
19301: 181 unit uPSI_JclConsole; //JCL
19302: 182 unit uPSI_JclLANMan; //JCL
19303: 183 unit uPSI_IdCustomHTTPServer; //Indy
19304: 184 unit IdHTTPServer; //Indy
19305: 185 unit uPSI_IdTCPServer; //Indy
19306: 186 unit uPSI_IdSocketHandle; //Indy
19307: 187 unit uPSI_IdIOHandlerSocket; //Indy
19308: 188 unit IdIOHandler; //Indy
19309: 189 unit uPSI_cutils; //Bloodshed
19310: 190 unit uPSI_BoldUtils; //boldsoft
19311: 191 unit uPSI_IdSimpleServer; //Indy
19312: 192 unit uPSI_IdSSLOpenSSL; //Indy
19313: 193 unit uPSI_IdMultipartFormData; //Indy
19314: 194 unit uPSI_SynURIOpener; //SynEdit
19315: 195 unit uPSI_PerlRegEx; //PCRE
19316: 196 unit uPSI_IdHeaderList; //Indy
19317: 197 unit uPSI_StFirst; //SysTools4
19318: 198 unit uPSI_JvCtrls; //JCL
19319: 199 unit uPSI_IdTrivialFTPBase; //Indy
19320: 200 unit uPSI_IdTrivialFTP; //Indy
19321: 201 unit uPSI_IdUDPBase; //Indy
19322: 202 unit uPSI_IdUDPClient; //Indy
19323: 203 unit uPSI_utypes; //for DMATH.DLL
19324: 204 unit uPSI_ShellAPI; //Borland
19325: 205 unit uPSI_IdRemoteCMDClient; //Indy
19326: 206 unit uPSI_IdRemoteCMDServer; //Indy
19327: 207 unit IdRexecServer; //Indy
19328: 208 unit IdRexec; (unit uPSI_IdRexec;) //Indy
19329: 209 unit IdUDPServer; //Indy
19330: 210 unit IdTimeUDPServer; //Indy
19331: 211 unit IdTimeServer; //Indy
19332: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
19333: 213 unit uPSI_IdIPWatch; //Indy
19334: 214 unit uPSI_IdIrcServer; //Indy
19335: 215 unit uPSI_IdMessageCollection; //Indy
19336: 216 unit uPSI_cPEM; //Fundamentals 4
19337: 217 unit uPSI_cFundamentUtils; //Fundamentals 4
19338: 218 unit uPSI_uwinplot; //DMATH
19339: 219 unit uPSI_xrtl_util_CPUUtils; //ExtendedRTL

```



```

19340: 220 unit uPSI_GR32_System; //Graphics32
19341: 221 unit uPSI_cFileUtils; //Fundamentals 4
19342: 222 unit uPSI_cDateTime; (timemachine) //Fundamentals 4
19343: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
19344: 224 unit uPSI_cRandom; //Fundamentals 4
19345: 225 unit uPSI_ueval; //DMath
19346: 226 unit uPSI_xrtl_net_URIUtils; //ExtendedRTL
19347: 227 unit xrtl_net_URIUtils; //ExtendedRTL
19348: 228 unit uPSI_uFFT; (FFT) //DMath
19349: 229 unit uPSI_DBXChannel; //Delphi
19350: 230 unit uPSI_DBXIndyChannel; //Delphi Indy
19351: 231 unit uPSI_xrtl_util_COMCat; //ExtendedRTL
19352: 232 unit uPSI_xrtl_util_StrUtils; //ExtendedRTL
19353: 233 unit uPSI_xrtl_util_VariantUtils; //ExtendedRTL
19354: 234 unit uPSI_xrtl_util_FileUtils; //ExtendedRTL
19355: 235 unit xrtl_util_Compat; //ExtendedRTL
19356: 236 unit uPSI_OleAuto; //Borland
19357: 237 unit uPSI_xrtl_util_COMUtils; //ExtendedRTL
19358: 238 unit uPSI_CmAdmCtl; //Borland
19359: 239 unit uPSI_ValEdit2; //VCL
19360: 240 unit uPSI_GR32; //Graphics32
19361: 241 unit uPSI_GR32_Image; //Graphics32
19362: 242 unit uPSI_xrtl_util_TimeUtils; //ExtendedRTL
19363: 243 unit uPSI_xrtl_util_TimeZone; //ExtendedRTL
19364: 244 unit uPSI_xrtl_util_TimeStamp; //ExtendedRTL
19365: 245 unit uPSI_xrtl_util_Map; //ExtendedRTL
19366: 246 unit uPSI_xrtl_util_Set; //ExtendedRTL
19367: 247 unit uPSI_CPortMonitor; //ComPort
19368: 248 unit uPSI_StIniStm; //SysTools4
19369: 249 unit uPSI_GR32_ExtImage; //Graphics32
19370: 250 unit uPSI_GR32_OrdinalMaps; //Graphics32
19371: 251 unit uPSI_GR32_Rasterizers; //Graphics32
19372: 252 unit uPSI_xrtl_util_Exception; //ExtendedRTL
19373: 253 unit uPSI_xrtl_util_Value; //ExtendedRTL
19374: 254 unit uPSI_xrtl_util_Compare; //ExtendedRTL
19375: 255 unit uPSI_FlatSB; //VCL
19376: 256 unit uPSI_JvAnalogClock; //JCL
19377: 257 unit uPSI_JvAlarms; //JCL
19378: 258 unit uPSI_JvSQLS; //JCL
19379: 259 unit uPSI_JvDBSecur; //JCL
19380: 260 unit uPSI_JvDBQBE; //JCL
19381: 261 unit uPSI_JvStarfield; //JCL
19382: 262 unit uPSI_JVCLMiscal; //JCL
19383: 263 unit uPSI_JvProfiler32; //JCL
19384: 264 unit uPSI_JvDirectories; //JCL
19385: 265 unit uPSI_JclSchedule; //JCL
19386: 266 unit uPSI_JclSvcCtrl; //JCL
19387: 267 unit uPSI_JvSoundControl; //JCL
19388: 268 unit uPSI_JvBDESQLScript; //JCL
19389: 269 unit uPSI_JvgDigits; //JCL>
19390: 270 unit uPSI_ImgList; //TCustomImageList
19391: 271 unit uPSI_JclMIDI; //JCL>
19392: 272 unit uPSI_JclWinMidi; //JCL>
19393: 273 unit uPSI_JclNTFS; //JCL>
19394: 274 unit uPSI_JclAppInst; //JCL>
19395: 275 unit uPSI_JvRle; //JCL>
19396: 276 unit uPSI_JvRas32; //JCL>
19397: 277 unit uPSI_JvImageDrawThread; //JCL>
19398: 278 unit uPSI_JvImageWindow; //JCL>
19399: 279 unit uPSI_JvTransparentForm; //JCL>
19400: 280 unit uPSI_JvWinDialogs; //JCL>
19401: 281 unit uPSI_JvSimLogic; //JCL>
19402: 282 unit uPSI_JvSimIndicator; //JCL>
19403: 283 unit uPSI_JvSimPID; //JCL>
19404: 284 unit uPSI_JvSimPIDLinker; //JCL>
19405: 285 unit uPSI_IdRFCReply; //Indy
19406: 286 unit uPSI_IdIdent; //Indy
19407: 287 unit uPSI_IdIdentServer; //Indy
19408: 288 unit uPSI_JvPatchFile; //JCL
19409: 289 unit uPSI_StNetPfm; //SysTools4
19410: 290 unit uPSI_StNet; //SysTools4
19411: 291 unit uPSI_JclPeImage; //JCL
19412: 292 unit uPSI_JclPrint; //JCL
19413: 293 unit uPSI_JclMime; //JCL
19414: 294 unit uPSI_JvRichEdit; //JCL
19415: 295 unit uPSI_JvDBRichEd; //JCL
19416: 296 unit uPSI_JvDice; //JCL
19417: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
19418: 298 unit uPSI_JvDirFrm; //JCL
19419: 299 unit uPSI_JvDualList; //JCL
19420: 300 unit uPSI_JvSwitch; ///JCL
19421: 301 unit uPSI_JvTimerLst; ///JCL
19422: 302 unit uPSI_JvMemTable; //JCL
19423: 303 unit uPSI_JvObjStr; //JCL
19424: 304 unit uPSI_StLArr; //SysTools4
19425: 305 unit uPSI_StWmDCpy; //SysTools4
19426: 306 unit uPSI_StText; //SysTools4
19427: 307 unit uPSI_StNTLog; //SysTools4
19428: 308 unit uPSI_xrtl_math_Integer; //ExtendedRTL

```

```

19429: 309 unit uPSI_JvImagPrvw; //JCL
19430: 310 unit uPSI_JvFormPatch; //JCL
19431: 311 unit uPSI_JvPicClip; //JCL
19432: 312 unit uPSI_JvDataConv; //JCL
19433: 313 unit uPSI_JvCpuUsage; //JCL
19434: 314 unit uPSI_JclUnitConv_mX2; //JCL
19435: 315 unit JvDualListForm; //JCL
19436: 316 unit uPSI_JvCpuUsage2; //JCL
19437: 317 unit uPSI_JvParserForm; //JCL
19438: 318 unit uPSI_JvJanTreeView; //JCL
19439: 319 unit uPSI_JvTransLED; //JCL
19440: 320 unit uPSI_JvPlaylist; //JCL
19441: 321 unit uPSI_JvFormAutoSize; //JCL
19442: 322 unit uPSI_JvYearGridEditForm; //JCL
19443: 323 unit uPSI_JvMarkupCommon; //JCL
19444: 324 unit uPSI_JvChart; //JCL
19445: 325 unit uPSI_JvXPCore; //JCL
19446: 326 unit uPSI_JvXPCoreUtils; //JCL
19447: 327 unit uPSI_StatsClasses; //mX4
19448: 328 unit uPSI_ExtCtrls2; //VCL
19449: 329 unit uPSI_JvUrlGrabbers; //JCL
19450: 330 unit uPSI_JvXmlTree; //JCL
19451: 331 unit uPSI_JvWavePlayer; //JCL
19452: 332 unit uPSI_JvUnicodeCanvas; //JCL
19453: 333 unit uPSI_JvTFUtils; //JCL
19454: 334 unit uPSI_IdServerIOHandler; //Indy
19455: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
19456: 336 unit uPSI_IdMessageCoder; //Indy
19457: 337 unit uPSI_IdMessageCoderMIME; //Indy
19458: 338 unit uPSI_IdMIMETypes; //Indy
19459: 339 unit uPSI_JvConverter; //JCL
19460: 340 unit uPSI_JvCsvParse; //JCL
19461: 341 unit uPSI_umath; unit uPSI_ugamma; //DMath
19462: 342 unit uPSI_ExcelExport; (Nat:TJsExcelExport) //JCL
19463: 343 unit uPSI_JvDBGridExport; //JCL
19464: 344 unit uPSI_JvgExport; //JCL
19465: 345 unit uPSI_JvSerialMaker; //JCL
19466: 346 unit uPSI_JvWin32; //JCL
19467: 347 unit uPSI_JvPaintFX; //JCL
19468: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
19469: 349 unit uPSI_JvValidators; (preview) //JCL
19470: 350 unit uPSI_JvNTEventLog; //JCL
19471: 351 unit uPSI_ShellZipTool; //mX4
19472: 352 unit uPSI_JvJoystick; //JCL
19473: 353 unit uPSI_JvMailSlots; //JCL
19474: 354 unit uPSI_JclComplex; //JCL
19475: 355 unit uPSI_SynPdf; //Synopse
19476: 356 unit uPSI_Registry; //VCL
19477: 357 unit uPSI_TlHelp32; //VCL
19478: 358 unit uPSI_JclRegistry; //JCL
19479: 359 unit uPSI_JvAirBrush; //JCL
19480: 360 unit uPSI_mORMotReport; //Synopse
19481: 361 unit uPSI_JclLocales; //JCL
19482: 362 unit uPSI_SynEdit; //SynEdit
19483: 363 unit uPSI_SynEditTypes; //SynEdit
19484: 364 unit uPSI_SynMacroRecorder; //SynEdit
19485: 365 unit uPSI_LongIntList; //SynEdit
19486: 366 unit uPSI_devutils; //DevC
19487: 367 unit uPSI_SynEditMiscClasses; //SynEdit
19488: 368 unit uPSI_SynEditRegexSearch; //SynEdit
19489: 369 unit uPSI_SynEditHighlighter; //SynEdit
19490: 370 unit uPSI_SynHighlighterPas; //SynEdit
19491: 371 unit uPSI_JvSearchFiles; //JCL
19492: 372 unit uPSI_SynHighlighterAny; //Lazarus
19493: 373 unit uPSI_SynEditKeyCmds; //SynEdit
19494: 374 unit uPSI_SynEditMiscProcs; //SynEdit
19495: 375 unit uPSI_SynEditKbdHandler; //SynEdit
19496: 376 unit uPSI_JvAppInst; //JCL
19497: 377 unit uPSI_JvAppEvent; //JCL
19498: 378 unit uPSI_JvAppCommand; //JCL
19499: 379 unit uPSI_JvAnimTitle; //JCL
19500: 380 unit uPSI_JvAnimatedImage; //JCL
19501: 381 unit uPSI_SynEditExport; //SynEdit
19502: 382 unit uPSI_SynExportHTML; //SynEdit
19503: 383 unit uPSI_SynExportRTF; //SynEdit
19504: 384 unit uPSI_SynEditSearch; //SynEdit
19505: 385 unit uPSI_fMain_back; //maXbox;
19506: 386 unit uPSI_JvZoom; //JCL
19507: 387 unit uPSI_PMrand; //PM
19508: 388 unit uPSI_JvSticker; //JCL
19509: 389 unit uPSI_XmlVerySimple; //mX4
19510: 390 unit uPSI_Services; //ExtPascal
19511: 391 unit uPSI_ExtPascalUtils; //ExtPascal
19512: 392 unit uPSI_SocketsDelphi; //ExtPascal
19513: 393 unit uPSI_StBarC; //SysTools
19514: 394 unit uPSI_StDbBarC; //SysTools
19515: 395 unit uPSI_StBarPN; //SysTools
19516: 396 unit uPSI_StDbPNBC; //SysTools
19517: 397 unit uPSI_StDb2DBC; //SysTools

```

```

19518: 398 unit uPSI_StMoney; //SysTools
19519: 399 unit uPSI_JvForth; //JCL
19520: 400 unit uPSI_RestRequest; //mX4
19521: 401 unit uPSI_HttpRESTConnectionIndy; //mX4
19522: 402 unit uPSI_JvXmlDatabase; //update
19523: 403 unit uPSI_StAstro; //SysTools
19524: 404 unit uPSI_StSort; //SysTools
19525: 405 unit uPSI_StDate; //SysTools
19526: 406 unit uPSI_StDateSt; //SysTools
19527: 407 unit uPSI_StBase; //SysTools
19528: 408 unit uPSI_StVInfo; //SysTools
19529: 409 unit uPSI_JvBrowseFolder; //JCL
19530: 410 unit uPSI_JvBoxProcs; //JCL
19531: 411 unit uPSI_urandom; (unit uranuvag;) //DMath
19532: 412 unit uPSI_usimann; (unit ugenalg;) //DMath
19533: 413 unit uPSI_JvHighlighter; //JCL
19534: 414 unit uPSI_Diff; //mX4
19535: 415 unit uPSI_SpringWinAPI; //DSpring
19536: 416 unit uPSI_StBits; //SysTools
19537: 417 unit uPSI_TomDBQue; //mX4
19538: 418 unit uPSI_MultilangTranslator; //mX4
19539: 419 unit uPSI_HyperLabel; //mX4
19540: 420 unit uPSI_Starter; //mX4
19541: 421 unit uPSI_FileAssocs; //devC
19542: 422 unit uPSI_devFileMonitorX; //devC
19543: 423 unit uPSI_devrun; //devC
19544: 424 unit uPSI_devExec; //devC
19545: 425 unit uPSI_oysUtils; //devC
19546: 426 unit uPSI_DosCommand; //devC
19547: 427 unit uPSI_CppTokenizer; //devC
19548: 428 unit uPSI_JvHLPParser; //devC
19549: 429 unit uPSI_JclMapi; //JCL
19550: 430 unit uPSI_JclShell; //JCL
19551: 431 unit uPSI_JclCOM; //JCL
19552: 432 unit uPSI_GR32_Math; //Graphics32
19553: 433 unit uPSI_GR32_LowLevel; //Graphics32
19554: 434 unit uPSI_SimpleHl; //mX4
19555: 435 unit uPSI_GR32_Filters; //Graphics32
19556: 436 unit uPSI_GR32_VectorMaps; //Graphics32
19557: 437 unit uPSI_cXMLFunctions; //Fundamentals 4
19558: 438 unit uPSI_JvTimer; //JCL
19559: 439 unit uPSI_cHTTPUtils; //Fundamentals 4
19560: 440 unit uPSI_cTLSUtils; //Fundamentals 4
19561: 441 unit uPSI_JclGraphics; //JCL
19562: 442 unit uPSI_JclSynch; //JCL
19563: 443 unit uPSI_IdTelnet; //Indy
19564: 444 unit uPSI_IdTelnetServer; //Indy
19565: 445 unit uPSI_IdEcho; //Indy
19566: 446 unit uPSI_IdEchoServer; //Indy
19567: 447 unit uPSI_IdEchoUDP; //Indy
19568: 448 unit uPSI_IdEchoUDPServer; //Indy
19569: 449 unit uPSI_IdSocks; //Indy
19570: 450 unit uPSI_IdAntiFreezeBase; //Indy
19571: 451 unit uPSI_IdHostnameServer; //Indy
19572: 452 unit uPSI_IdTunnelCommon; //Indy
19573: 453 unit uPSI_IdTunnelMaster; //Indy
19574: 454 unit uPSI_IdTunnelSlave; //Indy
19575: 455 unit uPSI_IdRSH; //Indy
19576: 456 unit uPSI_IdRSHServer; //Indy
19577: 457 unit uPSI_Spring_Cryptography_Utils; //Spring4Delphi
19578: 458 unit uPSI_MapReader; //devC
19579: 459 unit uPSI_LibTar; //devC
19580: 460 unit uPSI_IdStack; //Indy
19581: 461 unit uPSI_IdBlockCipherIntercept; //Indy
19582: 462 unit uPSI_IdChargenServer; //Indy
19583: 463 unit uPSI_IdFTPServer; //Indy
19584: 464 unit uPSI_IdException; //Indy
19585: 465 unit uPSI_utexplot; //DMath
19586: 466 unit uPSI_uwinstr; //DMath
19587: 467 unit uPSI_VarRecUtils; //devC
19588: 468 unit uPSI_JvStringListToHtml; //JCL
19589: 469 unit uPSI_JvStringHolder; //JCL
19590: 470 unit uPSI_IdCoder; //Indy
19591: 471 unit uPSI_SynHighlighterDfm; //Synedit
19592: 472 unit uHighlighterProcs; in 471 //Synedit
19593: 473 unit uPSI_LazFileUtils; //LCL
19594: 474 unit uPSI_IDECmdLine; //LCL
19595: 475 unit uPSI_lazMasks; //LCL
19596: 476 unit uPSI_ip_misc; //mX4
19597: 477 unit uPSI_Barcode; //LCL
19598: 478 unit uPSI_SimpleXML; //LCL
19599: 479 unit uPSI_JclIniFiles; //JCL
19600: 480 unit uPSI_D2XXUnit; {$X-} //FTDI
19601: 481 unit uPSI_JclDateTime; //JCL
19602: 482 unit uPSI_JclEDI; //JCL
19603: 483 unit uPSI_JclMiscel2; //JCL
19604: 484 unit uPSI_JclValidation; //JCL
19605: 485 unit uPSI_JclAnsiStrings; {-PString} //JCL
19606: 486 unit uPSI_SynEditMiscProcs2; //Synedit

```

```

19607: 487 unit uPSI_JclStreams; //JCL
19608: 488 unit uPSI_QRCode; //mX4
19609: 489 unit uPSI_BlockSocket; //ExtPascal
19610: 490 unit uPSI_Masks,Utills //VCL
19611: 491 unit uPSI_synauttil; //Synapse!
19612: 492 unit uPSI_JclMath_Class; //JCL RTL
19613: 493 unit ugamdist; //Gamma function //DMath
19614: 494 unit uibeta, ucorrel; //IBeta //DMath
19615: 495 unit uPSI_SRMgr; //mX4
19616: 496 unit uPSI_HotLog; //mX4
19617: 497 unit uPSI_DebugBox; //mX4
19618: 498 unit uPSI_ustrings; //DMath
19619: 499 unit uPSI_uregtest; //DMath
19620: 500 unit uPSI_usimplex; //DMath
19621: 501 unit uPSI_uhyper; //DMath
19622: 502 unit uPSI_IdHL7; //Indy
19623: 503 unit uPSI_IdIPMCastBase, //Indy
19624: 504 unit uPSI_IdIPMCastServer; //Indy
19625: 505 unit uPSI_IdIPMCastClient; //Indy
19626: 506 unit uPSI_unlfit; //nlregression //DMath
19627: 507 unit uPSI_IdRawHeaders; //Indy
19628: 508 unit uPSI_IdRawClient; //Indy
19629: 509 unit uPSI_IdRawFunctions; //Indy
19630: 510 unit uPSI_IdTCPStream; //Indy
19631: 511 unit uPSI_IdSNPP; //Indy
19632: 512 unit uPSI_St2DBarC; //SysTools
19633: 513 unit uPSI_ImageWin; //FTL //VCL
19634: 514 unit uPSI_CustomDrawTreeView; //FTL //VCL
19635: 515 unit uPSI_GraphWin; //FTL //VCL
19636: 516 unit uPSI_actionMain; //FTL //VCL
19637: 517 unit uPSI_StSpawn; //SysTools
19638: 518 unit uPSI_CtlPanel; //VCL
19639: 519 unit uPSI_IdLPR; //Indy
19640: 520 unit uPSI_SockRequestInterpreter; //Indy
19641: 521 unit uPSI_ulambert; //DMath
19642: 522 unit uPSI_ucholesk; //DMath
19643: 523 unit uPSI_SimpleDS; //VCL
19644: 524 unit uPSI_DBXSqlScanner; //VCL
19645: 525 unit uPSI_DBXMetaDataUtil; //VCL
19646: 526 unit uPSI_Chart; //TEE
19647: 527 unit uPSI_TeeProcs; //TEE
19648: 528 unit mXBDEUtils; //mX4
19649: 529 unit uPSI_MDIEdit; //VCL
19650: 530 unit uPSI_CopyPrsr; //VCL
19651: 531 unit uPSI_SockApp; //VCL
19652: 532 unit uPSI_AppEvnts; //VCL
19653: 533 unit uPSI_ExtActns; //VCL
19654: 534 unit uPSI_TeEngine; //TEE
19655: 535 unit uPSI_CoolMain; //browser //VCL
19656: 536 unit uPSI_StCRC; //SysTools
19657: 537 unit uPSI_StDecMth2; //SysTools
19658: 538 unit uPSI_frmExportMain; //Synedit
19659: 539 unit uPSI_SynDBEdit; //Synedit
19660: 540 unit uPSI_SynEditWildcardSearch; //Synedit
19661: 541 unit uPSI_BoldComUtils; //BOLD
19662: 542 unit uPSI_BoldIsoDateTime; //BOLD
19663: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
19664: 544 unit uPSI_BoldXMLRequests; //BOLD
19665: 545 unit uPSI_BoldStringList; //BOLD
19666: 546 unit uPSI_BoldFileHandler; //BOLD
19667: 547 unit uPSI_BoldContainers; //BOLD
19668: 548 unit uPSI_BoldQueryUserDlg; //BOLD
19669: 549 unit uPSI_BoldWinINet; //BOLD
19670: 550 unit uPSI_BoldQueue; //BOLD
19671: 551 unit uPSI_JvPcx; //JCL
19672: 552 unit uPSI_IdWhois; //Indy
19673: 553 unit uPSI_IdWhoIsServer; //Indy
19674: 554 unit uPSI_IdGopher; //Indy
19675: 555 unit uPSI_IdDateTimeStamp; //Indy
19676: 556 unit uPSI_IdDayTimeServer; //Indy
19677: 557 unit uPSI_IdDayTimeUDP; //Indy
19678: 558 unit uPSI_IdDayTimeUDPServer; //Indy
19679: 559 unit uPSI_IdDICTServer; //Indy
19680: 560 unit uPSI_IdDiscardServer; //Indy
19681: 561 unit uPSI_IdDiscardUDPServer; //Indy
19682: 562 unit uPSI_IdMappedFTP; //Indy
19683: 563 unit uPSI_IdMappedPortTCP; //Indy
19684: 564 unit uPSI_IdGopherServer; //Indy
19685: 565 unit uPSI_IdQotdServer; //Indy
19686: 566 unit uPSI_JvRgbToHtml; //JCL
19687: 567 unit uPSI_JvRemLog; //JCL
19688: 568 unit uPSI_JvSysComp; //JCL
19689: 569 unit uPSI_JvTMTL; //JCL
19690: 570 unit uPSI_JvWinampAPI; //JCL
19691: 571 unit uPSI_MSysUtils; //mX4
19692: 572 unit uPSI_ESBMaths; //ESB
19693: 573 unit uPSI_ESBMaths2; //ESB
19694: 574 unit uPSI_uLkJSON; //Lk
19695: 575 unit uPSI_ZURL; //Zeos //Zeos

```



```

19696: 576 unit uPSI_ZSysUtils; //Zeos
19697: 577 unit unaUtils internals //UNA
19698: 578 unit uPSI_ZMatchPattern; //Zeos
19699: 579 unit uPSI_ZClasses; //Zeos
19700: 580 unit uPSI_ZCollections; //Zeos
19701: 581 unit uPSI_ZEncoding; //Zeos
19702: 582 unit uPSI_IdRawBase; //Indy
19703: 583 unit uPSI_IdNTLM; //Indy
19704: 584 unit uPSI_IdNNTP; //Indy
19705: 585 unit uPSI_usniffer; //PortScanForm
19706: 586 unit uPSI_IdCoderMIME; //Indy
19707: 587 unit uPSI_IdCoderUUE; //Indy
19708: 588 unit uPSI_IdCoderXXE; //Indy
19709: 589 unit uPSI_IdCoder3to4; //Indy
19710: 590 unit uPSI_IdCookie; //Indy
19711: 591 unit uPSI_IdCookieManager; //Indy
19712: 592 unit uPSI_WDosSocketUtils; //WDos
19713: 593 unit uPSI_WDosPlcUtils; //WDos
19714: 594 unit uPSI_WDosPorts; //WDos
19715: 595 unit uPSI_WDosResolvers; //WDos
19716: 596 unit uPSI_WDosTimers; //WDos
19717: 597 unit uPSI_WDosPlcs; //WDos
19718: 598 unit uPSI_WDosPneumatics; //WDos
19719: 599 unit uPSI_IdFingerServer; //Indy
19720: 600 unit uPSI_IdDNSResolver; //Indy
19721: 601 unit uPSI_IdHTTPWebBrokerBridge; //Indy
19722: 602 unit uPSI_IdIntercept; //Indy
19723: 603 unit uPSI_IdIPMCastBase; //Indy
19724: 604 unit uPSI_IdLogBase; //Indy
19725: 605 unit uPSI_IdIOHandlerStream; //Indy
19726: 606 unit uPSI_IdMappedPortUDP; //Indy
19727: 607 unit uPSI_IdQOTDUDPServer; //Indy
19728: 608 unit uPSI_IdQOTDUDP; //Indy
19729: 609 unit uPSI_IdSysLog; //Indy
19730: 610 unit uPSI_IdSysLogServer; //Indy
19731: 611 unit uPSI_IdSysLogMessage; //Indy
19732: 612 unit uPSI_IdTimeServer; //Indy
19733: 613 unit uPSI_IdTimeUDP; //Indy
19734: 614 unit uPSI_IdTimeUDPServer; //Indy
19735: 615 unit uPSI_IdUserAccounts; //Indy
19736: 616 unit uPSI_TextUtils; //mX4
19737: 617 unit uPSI_MandelbrotEngine; //mX4
19738: 618 unit uPSI_delphi_arduino_Unit1; //mX4
19739: 619 unit uPSI_DTDSchema2; //mX4
19740: 620 unit uPSI_fplotMain; //DMath
19741: 621 unit uPSI_FindFileIter; //mX4
19742: 622 unit uPSI_PppState; (JclStrHashMap) //PPP
19743: 623 unit uPSI_PppParser; //PPP
19744: 624 unit uPSI_PppLexer; //PPP
19745: 625 unit uPSI_PCharUtils; //PPP
19746: 626 unit uPSI_uJSON; //WU
19747: 627 unit uPSI_JclStrHashMap; //JCL
19748: 628 unit uPSI_JclHookExcept; //JCL
19749: 629 unit uPSI_EncdDecd; //VCL
19750: 630 unit uPSI_SockAppReg; //VCL
19751: 631 unit uPSI_PJFileHandle; //PJ
19752: 632 unit uPSI_PJEnvVars; //PJ
19753: 633 unit uPSI_PJPipe; //PJ
19754: 634 unit uPSI_PJPipeFilters; //PJ
19755: 635 unit uPSI_PJConsoleApp; //PJ
19756: 636 unit uPSI_UConsoleAppEx; //PJ
19757: 637 unit uPSI_DbxSocketChannelNative; //VCL
19758: 638 unit uPSI_DbxDataGenerator; //VCL
19759: 639 unit uPSI_DBXClient; //VCL
19760: 640 unit uPSI_IdLogEvent; //Indy
19761: 641 unit uPSI_Reversi; //mX4
19762: 642 unit uPSI_Geometry; //mX4
19763: 643 unit uPSI_IdSMTPServer; //Indy
19764: 644 unit uPSI_Textures; //mX4
19765: 645 unit uPSI_IBX; //VCL
19766: 646 unit uPSI_IWDBCommon; //VCL
19767: 647 unit uPSI_SortGrid; //mX4
19768: 648 unit uPSI_IB; //VCL
19769: 649 unit uPSI_IBScript; //VCL
19770: 650 unit uPSI_JvCSVBaseControls; //JCL
19771: 651 unit uPSI_Jvg3DColors; //JCL
19772: 652 unit uPSI_JvHLEditor; //beat
19773: 653 unit uPSI_JvShellHook; //JCL
19774: 654 unit uPSI_DBCommon2; //VCL
19775: 655 unit uPSI_JvSHFileOperation; //JCL
19776: 656 unit uPSI_uFileexport; //mX4
19777: 657 unit uPSI_JvDialogs; //JCL
19778: 658 unit uPSI_JvDBTreeView; //JCL
19779: 659 unit uPSI_JvDBUltimGrid; //JCL
19780: 660 unit uPSI_JvDBQueryParamsForm; //JCL
19781: 661 unit uPSI_JvExControls; //JCL
19782: 662 unit uPSI_JvBDEMemTable; //JCL
19783: 663 unit uPSI_JvCommStatus; //JCL
19784: 664 unit uPSI_JvMailSlots2; //JCL

```

```

19785: 665 unit uPSI_JvgWinMask; //JCL
19786: 666 unit uPSI_StEclipse; //SysTools
19787: 667 unit uPSI_StMime; //SysTools
19788: 668 unit uPSI_StList; //SysTools
19789: 669 unit uPSI_StMerge; //SysTools
19790: 670 unit uPSI_StStrS; //SysTools
19791: 671 unit uPSI_StTree; //SysTools
19792: 672 unit uPSI_StVArr; //SysTools
19793: 673 unit uPSI_StRegIni; //SysTools
19794: 674 unit uPSI_urkf; //DMath
19795: 675 unit uPSI_usvd; //DMath
19796: 676 unit uPSI_DepWalkUtils; //JCL
19797: 677 unit uPSI_OptionsFrm; //JCL
19798: 678 unit yuvconverts; //mX4
19799: 679 uPSI_JvPropAutoSave; //JCL
19800: 680 uPSI_AcLAPI; //alcinoe
19801: 681 uPSI_AviCap; //alcinoe
19802: 682 uPSI_ALAVLBinaryTree; //alcinoe
19803: 683 uPSI_ALFcnMisc; //alcinoe
19804: 684 uPSI_ALStringList; //alcinoe
19805: 685 uPSI_ALQuickSortList; //alcinoe
19806: 686 uPSI_ALStaticText; //alcinoe
19807: 687 uPSI_ALJSONDoc; //alcinoe
19808: 688 uPSI_ALGSMComm; //alcinoe
19809: 689 uPSI_ALWindows; //alcinoe
19810: 690 uPSI_ALMultiPartFormDataParser; //alcinoe
19811: 691 uPSI_ALHttpCommon; //alcinoe
19812: 692 uPSI_ALWebSpider; //alcinoe
19813: 693 uPSI_ALHttpClient; //alcinoe
19814: 694 uPSI_ALFcnHTML; //alcinoe
19815: 695 uPSI_ALFTPClient; //alcinoe
19816: 696 uPSI_ALInternetMessageCommon; //alcinoe
19817: 697 uPSI_ALWininetHttpClient; //alcinoe
19818: 698 uPSI_ALWininetFTPClient; //alcinoe
19819: 699 uPSI_ALWinHttpWrapper; //alcinoe
19820: 700 uPSI_ALWinHttpClient; //alcinoe
19821: 701 uPSI_ALFcnWinSock; //alcinoe
19822: 702 uPSI_ALFcnSQL; //alcinoe
19823: 703 uPSI_ALFcnCGI; //alcinoe
19824: 704 uPSI_ALFcnExecute; //alcinoe
19825: 705 uPSI_ALFcnFile; //alcinoe
19826: 706 uPSI_ALFcnMime; //alcinoe
19827: 707 uPSI_ALPhpRunner; //alcinoe
19828: 708 uPSI_ALGraphic; //alcinoe
19829: 709 uPSI_ALIniFiles; //alcinoe
19830: 710 uPSI_ALMemCachedClient; //alcinoe
19831: 711 unit uPSI_MyGrids; //mX4
19832: 712 uPSI_ALMultiPartMixedParser; //alcinoe
19833: 713 uPSI_ALSMTPClient; //alcinoe
19834: 714 uPSI_ALNNTTPClient; //alcinoe
19835: 715 uPSI_ALHintBalloon; //alcinoe
19836: 716 unit uPSI_ALXmlDoc; //alcinoe
19837: 717 unit uPSI_IPCThrd; //VCL
19838: 718 unit uPSI_MonForm; //VCL
19839: 719 unit uPSI_TeCanvas; //Orpheus
19840: 720 unit uPSI_Ovcmisc; //Orpheus
19841: 721 unit uPSI_ovcfiler; //Orpheus
19842: 722 unit uPSI_ovcstate; //Orpheus
19843: 723 unit uPSI_ovccoco; //Orpheus
19844: 724 unit uPSI_ovcrvexp; //Orpheus
19845: 725 unit uPSI_OvcFormatSettings; //Orpheus
19846: 726 unit uPSI_OvcUtils; //Orpheus
19847: 727 unit uPSI_ovcstore; //Orpheus
19848: 728 unit uPSI_ovcstr; //Orpheus
19849: 729 unit uPSI_ovcmru; //Orpheus
19850: 730 unit uPSI_ovccmd; //Orpheus
19851: 731 unit uPSI_ovctimer; //Orpheus
19852: 732 unit uPSI_ovcintl; //Orpheus
19853: 733 uPSI_AfCircularBuffer; //AsyncFree
19854: 734 uPSI_AfUtils; //AsyncFree
19855: 735 uPSI_AfSafeSync; //AsyncFree
19856: 736 uPSI_AfComPortCore; //AsyncFree
19857: 737 uPSI_AfComPort; //AsyncFree
19858: 738 uPSI_AfPortControls; //AsyncFree
19859: 739 uPSI_AfDataDispatcher; //AsyncFree
19860: 740 uPSI_AfViewers; //AsyncFree
19861: 741 uPSI_AfDataTerminal; //AsyncFree
19862: 742 uPSI_SimplePortMain; //AsyncFree
19863: 743 unit uPSI_ovcclock; //Orpheus
19864: 744 unit uPSI_o32intlst; //Orpheus
19865: 745 unit uPSI_o32ledlabel; //Orpheus
19866: 746 unit uPSI_ALMySQLClient; //alcinoe
19867: 747 unit uPSI_ALFBXClient; //alcinoe
19868: 748 unit uPSI_ALFcnSQL; //alcinoe
19869: 749 unit uPSI_AsyncTimer; //mX4
19870: 750 unit uPSI_ApplicationFileIO; //mX4
19871: 751 unit uPSI_PsAPI; //VCLÉ
19872: 752 uPSI_ovcuser; //Orpheus
19873: 753 uPSI_ovcurl; //Orpheus

```

```

19874: 754 uPSI_ovcvlb; //Orpheus
19875: 755 uPSI_ovccolor; //Orpheus
19876: 756 uPSI_ALFBXLib; //alcinoe
19877: 757 uPSI_ovcmeter; //Orpheus
19878: 758 uPSI_ovcpeakm; //Orpheus
19879: 759 uPSI_O32BGSty; //Orpheus
19880: 760 uPSI_ovcBidi; //Orpheus
19881: 761 uPSI_ovctcary; //Orpheus
19882: 762 uPSI_DXPUTils; //mX4
19883: 763 uPSI_ALMultiPartBaseParser; //alcinoe
19884: 764 uPSI_ALMultiPartAlternativeParser; //alcinoe
19885: 765 uPSI_ALPOP3Client; //alcinoe
19886: 766 uPSI_SmallUtils; //mX4
19887: 767 uPSI_MakeApp; //mX4
19888: 768 uPSI_O32MouseMon; //Orpheus
19889: 769 uPSI_OvcCache; //Orpheus
19890: 770 uPSI_ovccalc; //Orpheus
19891: 771 uPSI_Joystick; //OpenGL
19892: 772 uPSI_ScreenSaver; //OpenGL
19893: 773 uPSI_XCollection; //OpenGL
19894: 774 uPSI_Polynomials; //OpenGL
19895: 775 uPSI_PersistentClasses, //9.86 //OpenGL
19896: 776 uPSI_VectorLists; //OpenGL
19897: 777 uPSI_XOpenGL; //OpenGL
19898: 778 uPSI_MeshUtils; //OpenGL
19899: 779 unit uPSI_JclSysUtils; //JCL
19900: 780 unit uPSI_JclBorlandTools; //JCL
19901: 781 unit JclFileUtils_max; //JCL
19902: 782 uPSI_AfDataControls; //AsyncFree
19903: 783 uPSI_GLSilhouette; //OpenGL
19904: 784 uPSI_JclSysUtils_class; //JCL
19905: 785 uPSI_JclFileUtils_class; //JCL
19906: 786 uPSI_FileUtil; //JCL
19907: 787 uPSI_changefind; //mX4
19908: 788 uPSI_cmdIntf; //mX4
19909: 789 uPSI_fsrvic; //mX4
19910: 790 uPSI_Keyboard; //OpenGL
19911: 791 uPSI_VRMLParser; //OpenGL
19912: 792 uPSI_GLFileVRML; //OpenGL
19913: 793 uPSI_Octree; //OpenGL
19914: 794 uPSI_GLPolyhedron; //OpenGL
19915: 795 uPSI_GLCrossPlatform; //OpenGL
19916: 796 uPSI_GLParticles; //OpenGL
19917: 797 uPSI_GLNavigator; //OpenGL
19918: 798 uPSI_GLStarRecord; //OpenGL
19919: 799 uPSI_GLTextureCombiners; //OpenGL
19920: 800 uPSI_GLCanvas; //OpenGL
19921: 801 uPSI_GeometryBB; //OpenGL
19922: 802 uPSI_GeometryCoordinates; //OpenGL
19923: 803 uPSI_VectorGeometry; //OpenGL
19924: 804 uPSI_BumpMapping; //OpenGL
19925: 805 uPSI_TGA; //OpenGL
19926: 806 uPSI_GLVectorFileObjects; //OpenGL
19927: 807 uPSI_IMM; //VCL
19928: 808 uPSI_CategoryButtons; //VCL
19929: 809 uPSI_ButtonGroup; //VCL
19930: 810 uPSI_DbExcept; //VCL
19931: 811 uPSI_AxCtrls; //VCL
19932: 812 uPSI_GL_actorUnit1; //OpenGL
19933: 813 uPSI_StdVCL; //VCL
19934: 814 unit CurvesAndSurfaces; //OpenGL
19935: 815 uPSI_DataAwareMain; //AsyncFree
19936: 816 uPSI_TabNotBk; //VCL
19937: 817 uPSI_udwsfiler; //mX4
19938: 818 uPSI_synaip; //Synapse!
19939: 819 uPSI_synacode; //Synapse
19940: 820 uPSI_synachar; //Synapse
19941: 821 uPSI_synamisc; //Synapse
19942: 822 uPSI_synaser; //Synapse
19943: 823 uPSI_synaicnv; //Synapse
19944: 824 uPSI_tlntsend; //Synapse
19945: 825 uPSI_pingsend; //Synapse
19946: 826 uPSI_blksock; //Synapse
19947: 827 uPSI_asnlutil; //Synapse
19948: 828 uPSI_dnssend; //Synapse
19949: 829 uPSI_clamsend; //Synapse
19950: 830 uPSI_ldapsend; //Synapse
19951: 831 uPSI_mimemess; //Synapse
19952: 832 uPSI_slogsend; //Synapse
19953: 833 uPSI_mimepart; //Synapse
19954: 834 uPSI_mimeinln; //Synapse
19955: 835 uPSI_ftpsend; //Synapse
19956: 836 uPSI_ftptsend; //Synapse
19957: 837 uPSI_httpsend; //Synapse
19958: 838 uPSI_sntpsend; //Synapse
19959: 839 uPSI_smtptsend; //Synapse
19960: 840 uPSI_snmptsend; //Synapse
19961: 841 uPSI_imapsend; //Synapse
19962: 842 uPSI_pop3send; //Synapse

```

```

19963: 843 uPSI_nntpsend; //Synapse
19964: 844 uPSI_ssl_cryptlib; //Synapse
19965: 845 uPSI_ssl_openssl; //Synapse
19966: 846 uPSI_synhttp_daemon; //Synapse
19967: 847 uPSI_NetWork; //mX4
19968: 848 uPSI_PingThread; //Synapse
19969: 849 uPSI_JvThreadTimer; //JCL
19970: 850 unit uPSI_wwSystem; //InfoPower
19971: 851 unit uPSI_IdIOComponent; //Indy
19972: 852 unit uPSI_IdIOHandlerThrottle; //Indy
19973: 853 unit uPSI_Themes; //VCL
19974: 854 unit uPSI_StdStyleActnCtrls; //VCL
19975: 855 unit uPSI_UDDIHelper; //VCL
19976: 856 unit uPSI_IdIMAP4Server; //Indy
19977: 857 uPSI_VariantSymbolTable, //VCL //3.9.9.92
19978: 858 uPSI_udf_glob, //mX4
19979: 859 uPSI_TabGrid, //VCL
19980: 860 uPSI_JsDBTreeView, //mX4
19981: 861 uPSI_JsSendMail, //mX4
19982: 862 uPSI_dbTvRecordList, //mX4
19983: 863 uPSI_TreeVwEx, //mX4
19984: 864 uPSI_ECDataLink, //mX4
19985: 865 uPSI_dbTree, //mX4
19986: 866 uPSI_dbTreeCBox, //mX4
19987: 867 unit uPSI_Debug; //TfrmDebug
19988: 868 uPSI_TimeFnct; //mX4
19989: 869 uPSI_FileIntf, //VCL
19990: 870 uPSI_SockTransport, //RTL
19991: 871 unit uPSI_WinInet; //RTL
19992: 872 unit uPSI_Wwstr; //mX4
19993: 873 uPSI_DBLookup, //VCL
19994: 874 uPSI_Hotspot, //mX4
19995: 875 uPSI_HList; //History List
19996: 876 unit uPSI_DrTable; //VCL
19997: 877 uPSI_TConnect, //VCL
19998: 978 uPSI_DataBkr, //VCL
19999: 979 uPSI_HTTPIntr; //VCL
20000: 980 unit uPSI_Mathbox; //mX4
20001: 881 uPSI_cyIndy, //cY
20002: 882 uPSI_cySysUtils, //cY
20003: 883 uPSI_cyWinUtils, //cY
20004: 884 uPSI_cyStrUtils, //cY
20005: 885 uPSI_cyObjUtils, //cY
20006: 886 uPSI_cyDateUtils, //cY
20007: 887 uPSI_cyBDE, //cY
20008: 888 uPSI_cyClasses, //cY
20009: 889 uPSI_cyGraphics, //3.9.9.94_2 //cY
20010: 890 unit uPSI_cyTypes; //cY
20011: 891 uPSI_JvDateTimePicker, //JCL
20012: 892 uPSI_JvCreateProcess, //JCL
20013: 893 uPSI_JvEasterEgg, //JCL
20014: 894 uPSI_WinSvc, //3.9.9.94_3 //VCL
20015: 895 uPSI_SvcMgr, //VCL
20016: 896 unit uPSI_JvPickDate; //JCL
20017: 897 unit uPSI_JvNotify; //JCL
20018: 898 uPSI_JvStrHlder, //JCL
20019: 899 unit uPSI_JclNTFS2; //JCL
20020: 900 uPSI_Jcl8087 //math coprocessor //JCL
20021: 901 uPSI_JvAddPrinter, //JCL
20022: 902 uPSI_JvCabFile, //JCL
20023: 903 uPSI_JvDataEmbedded; //JCL
20024:
20025:
20026:
20027: //////////////////////////////////////
20028: //Form Template Library FTL
20029: //////////////////////////////////////
20030:
20031: 26 FTL For Form Building out of the Script, eg. 399_form_templates.txt
20032:
20033: 045 unit uPSI_VListView TFormListView;
20034: 263 unit uPSI_JvProfiler32; TProfReport
20035: 270 unit uPSI_ImgList; TCustomImageList
20036: 278 unit uPSI_JvImageWindow; TJvImageWindow
20037: 317 unit uPSI_JvParserForm; TJvHTMLParserForm
20038: 497 unit uPSI_DebugBox; TDebugBox
20039: 513 unit uPSI_ImageWin; TImageForm, TImageForm2
20040: 514 unit uPSI_CustomDrawTreeView; TCustomDrawForm
20041: 515 unit uPSI_GraphWin; TGraphWinForm
20042: 516 unit uPSI_actionMain; TActionForm
20043: 518 unit uPSI_CtlPanel; TAppletApplication
20044: 529 unit uPSI_MDIEdit; TEditForm
20045: 535 unit uPSI_CoolMain; {browser} TWebMainForm
20046: 538 unit uPSI_frmExportMain; TSynexportForm
20047: 585 unit uPSI_usniffer; {PortScanForm} TSniffForm
20048: 600 unit uPSI_ThreadForm; TThreadSortForm;
20049: 618 unit uPSI_delphi_arduino_Unit1; TLEDForm
20050: 620 unit uPSI_fplotMain; TfplotForm1
20051: 660 unit uPSI_JvDBQueryParamsForm; TJvQueryParamsDialog

```



```

20052: 677 unit uPSI_OptionsFrm;                                TfrmOptions;
20053: 718 unit uPSI_MonForm;                                    TMonitorForm
20054: 742 unit uPSI_SimplePortMain;                             TPortForm1
20055: 770 unit uPSI_ovccalc;                                    TOvcCalculator //widget
20056: 810 unit uPSI_DbExcept;                                   TDbEngineErrorDlg
20057: 812 unit uPSI_GL_actorUnit1;                             TglActorForm1 //OpenGL Robot
20058: 846 unit uPSI_synhttp_daemon;                             TTCPHttpDaemon, TTCPHttpThrd, TPingThread
20059: 867 unit uPSI_Debug;                                       TfrmDebug
20060:
20061:
20062: ex.:with TEditForm.create(self) do begin
20063:     caption:= 'Template Form Tester';
20064:     FormStyle:= fsStayOnTop;
20065:     with editor do begin
20066:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
20067:         SelStart:= 0;
20068:         Modified:= False;
20069:     end;
20070: end;
20071: with TWebMainForm.create(self) do begin
20072:     URLs.Text:= 'http://www.kleiner.ch';
20073:     URLs.Click(self); Show;
20074: end;
20075: with TSynexportForm.create(self) do begin
20076:     Caption:= 'Synexport HTML RTF tester';
20077:     Show;
20078: end;
20079: with TThreadSortForm.create(self) do begin
20080:     showmodal; free;
20081: end;
20082:
20083: with TCustomDrawForm.create(self) do begin
20084:     width:=820; height:=820;
20085:     image1.height:= 600; //add properties
20086:     image1.picture.bitmap:= image2.picture.bitmap;
20087:     //SelectionBackground1Click(self) CustomDraw1Click(self);
20088:     Background1.click;
20089:     bitmap1.click;
20090:     Tile1.click;
20091:     Showmodal;
20092:     Free;
20093: end;
20094:
20095: with TfplotForm1.Create(self) do begin
20096:     BtnPlotClick(self);
20097:     Showmodal; Free;
20098: end;
20099:
20100: with TOvcCalculator.create(self) do begin
20101:     parent:= aForm;
20102:     //free;
20103:     setbounds(550,510,200,150);
20104:     displaystr:= 'maXcalc';
20105: end;
20106:
20107:
20108: //////////////////////////////////////
20109: All maXbox Tutorials Table of Content 2014
20110: //////////////////////////////////////
20111: Tutorial 00 Function-Coding (Blix the Programmer)
20112: Tutorial 01 Procedural-Coding
20113: Tutorial 02 OO-Programming
20114: Tutorial 03 Modular Coding
20115: Tutorial 04 UML Use Case Coding
20116: Tutorial 05 Internet Coding
20117: Tutorial 06 Network Coding
20118: Tutorial 07 Game Graphics Coding
20119: Tutorial 08 Operating System Coding
20120: Tutorial 09 Database Coding
20121: Tutorial 10 Statistic Coding
20122: Tutorial 11 Forms Coding
20123: Tutorial 12 SQL DB Coding
20124: Tutorial 13 Crypto Coding
20125: Tutorial 14 Parallel Coding
20126: Tutorial 15 Serial RS232 Coding
20127: Tutorial 16 Event Driven Coding
20128: Tutorial 17 Web Server Coding
20129: Tutorial 18 Arduino System Coding
20130: Tutorial 18_3 RGB LED System Coding
20131: Tutorial 19 WinCOM /Arduino Coding
20132: Tutorial 20 Regular Expressions RegEx
20133: Tutorial 21 Android Coding (coming 2013)
20134: Tutorial 22 Services Programming
20135: Tutorial 23 Real Time Systems
20136: Tutorial 24 Clean Code
20137: Tutorial 25 maXbox Configuration I+II
20138: Tutorial 26 Socket Programming with TCP
20139: Tutorial 27 XML & TreeView
20140: Tutorial 28 DLL Coding (available)

```

```
20141: Tutorial 29 UML Scripting (2014)
20142: Tutorial 30 Web of Things (coming 2014)
20143: Tutorial 31 Closures (coming 2014)
20144: Tutorial 32 SQL Firebird (coming 2014)
20145:
20146:
20147: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
20148: using Docu for this file is maxbox_functions_all.pdf
20149: PEP - Pascal Education Program Lib Lab
20150:
20151: http://stackoverflow.com/tags/pascalscript/hot
20152: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
20153: #locs:15162
20154: http://sourceforge.net/apps/mediawiki/maXbox
20155: http://www.blaisepascal.eu/
20156: https://github.com/maxkleiner/maXbox3.git
20157: http://www.heise.de/download/maxbox-1176464.html
20158: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
20159: https://www.facebook.com/pages/Programming-maXbox/166844836691703
20160:
20161: ---- bigbitbox code_cleared_checked----
```