

```

1: ****
2: Constructor Function and Procedure List of maxbox 3.9.9.195
3: ****
4:
5: //////////////////////////////////////////////////////////////////
6: ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX3
7: -----
8:
9: File EXE: 25152000 V3.9.9.195 Mai 2015 to EKON/BASTA/JAX/IBZ/SWS/EU/DT/PASCON
10: *****Now the Funclist*****
11: Funclist Function : 15031 //10766 //10165 (7648)
12: *****Now the Proclist*****
13: Proclist Procedure Size is: 9172 //6792 //6401 4752
14: *****Now Constructors*****
15: Constructlist Constructor Size is: 1492 //995 //
16: def head:max: maxBox7: 12.05.2015 16:41:55
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: 25695! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 25670
22: ASize of EXE: 25152000 (23614464) (22041600) (16586240) (13511680)
23: SHA1 Hash of maxbox 3.9.9.195: F0AB7D054111F5CE46BA122D6280397A841C6FAB
24:
25: -----
26: -----
27: //////////////////////////////////////////////////////////////////
28:
29:
30: FUNCTION Metric of Script: 256_findfunctions2_of_EXE.txt
31: Function *****Now the Funclist*****
32: function GetResStringChecked(Ident: string; const Args: array of const): string
33: Function ( Index: Longint): Integer
34: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
35: Function _CheckAutoResult( ResultCode: HResult) : HResult
36: function _T(Name: tbtString): Variant;
37: function ABNFToText(const AText : String) : String
38: Function Abs(e : Extended) : Extended;
39: Function Ackermann( const A, B : Integer) : Integer
40: Function AcquireLayoutLock : Boolean
41: Function ActionByName( const AName : string) : TWebActionItem
42: Function ACTIVEBUFFER : PCHAR
43: Function Add : TAggregate
44: function Add : TCollectionItem
45: Function Add : TColumn
46: Function Add : TComboExItem
47: Function Add : TCookie
48: Function Add : TCoolBand
49: Function Add : TFavoriteLinkItem
50: Function Add : TFileTypeItem
51: Function Add : THeaderSection
52: Function Add : THTMLTableColumn
53: Function Add : TIdEmailAddressItem
54: Function Add : TIdMessagePart
55: Function Add : TIdUserAccount
56: Function Add : TListColumn
57: Function Add : TListItem
58: Function Add : TStatusPanel
59: Function Add : TTaskDialogBaseButtonItem
60: Function Add : TWebActionItem
61: Function Add : TWorkArea
62: Function Add( AClass : TClass) : Integer
63: Function Add( AComponent : TComponent) : Integer
64: Function Add( AItem, AData : Integer) : Integer
65: Function Add( AItem, AData : Pointer) : Pointer
66: Function Add( AItem, AData : TObject) : TObject
67: Function Add( AObject : TObject) : Integer
68: Function Add( const Access, Count : Cardinal; const Offset : Int64) : Integer
69: Function Add( const S : WideString) : Integer
70: Function Add( Image, Mask : TBitmap) : Integer
71: Function Add( Index : LongInt; const Text : string) : LongInt
72: Function Add( Sibling : TTreeNode; const S : string) : TTreeNode
73: Function Add(const S: string): Integer
74: function Add(S: string): Integer;
75: Function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer) : Integer
76: Function ADDCHILD : TFIELDDEF
77: Function AddChild( Index : LongInt; const Text : string) : LongInt
78: Function AddChild( Parent : TTreeNode; const S : string) : TTreeNode
79: Function AddChildFirst( Parent : TTreeNode; const S : string) : TTreeNode
80: Function AddChildObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
81: Function AddChildObject( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
82: Function AddChildObjectFirst( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
83: Function ADDFIELDDEF : TFIELDDEF
84: Function AddFileExtIfNecessary( AFileName, AExt : string) : string
85: Function AddFirst( Sibling : TTreeNode; const S : string) : TTreeNode
86: Function AddIcon( Image : TIcon) : Integer
87: Function AddImage( Value : TCustomImageList; Index : Integer) : Integer
88: Function ADDINDEXDEF : TINDEXDEF
89: Function AddItem(const Caption:String;const ImageIdx,SelectImageIdx,OverlayImagIdx,
    Indent:Int;Data:Ptr):TComboExItem

```

```

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

```

```

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

```

```

268: Function BooleanToInteger( const Pb : Boolean) : Integer
269: Function BoolToStr(B: Boolean; UseBoolStrs: Boolean): string
270: Function BoolToStrl(value : boolean) : string;
271: Function booltoint( aBool : Boolean) : LongInt
272: Function inttobool( aInt : LongInt) : Boolean
273: Function Bounds( ALeft, ATop, AWidth, AHeight : Integer) : TRect
274: function Bounds(ALeft, ATop, AWidth, AHeight: Integer): TRect)
275: Function BreakApart( BaseString, BreakString : string; StringList : TStrings) : TStrings
276: Function BrightColor( const Color : TColor; const Pct : Single) : TColor
277: Function BrightColorChannel( const Channel : Byte; const Pct : Single) : Byte
278: Function BufferRequest( Length : Integer) : TStream
279: Function BuildfileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
280: Function Buttons : PTaskDialogButton
281: Function BytesPerScanline( PixelsPerScanline, BitsPerPixel, Alignment : Longint) : Longint
282: Function BytesToCardinal( const AValue : TIddBytes; const AIIndex : Integer) : Cardinal
283: Function BytesToChar( const AValue : TIddBytes; const AIIndex : Integer) : Char
284: Function BytesToInt64( const AValue : TIddBytes; const AIIndex : Integer) : Int64
285: Function BytesToInteger( const AValue : TIddBytes; const AIIndex : Integer) : Integer
286: Function BytesToIPv6( const AValue : TIddBytes; const AIIndex : Integer) : TIidIpv6Address
287: Function BytesToShort( const AValue : TIddBytes; const AIIndex : Integer) : Short
288: Function BytesToString(ABytes:TIdBytes; AstartIndex:Integer; AMaxCount:Integer): string;
289: Function BytesToStr(const Value : TBytes): String;
290: Function BytesToWord( const AValue : TIddBytes; const AIIndex : Integer) : Word
291: Function ByteToBin(Int: Byte): String;
292: Function ByteToCharIndex( S : string; Index : Integer) : Integer
293: function ByteToCharIndex(const S: string; Index: Integer): Integer
294: Function ByteToCharLen( S : string; MaxLen : Integer) : Integer
295: function ByteToCharLen(const S: string; MaxLen: Integer): Integer
296: Function ByteToHex( const AByte : Byte) : string
297: Function ByteToOctal( const AByte : Byte) : string
298: Function ByteType( S : string; Index : Integer) : TMbcsByteType
299: function ByteType(const S: string; Index: Integer): TMbcsByteType)
300: Function CalctitleRect( Col : TColumn; ARow : Integer; var MasterCol : TColumn) : TRect
301: Function CalculateDFAFingerprint( oStates : TList) : integer
302: function CallTerminateProcs: Boolean)
303: function CANFOCUS:BOOLEAN
304: Function CanLoad( const Ext : string) : Boolean
305: Function CanParse( AWebRequest : TWebRequest) : Boolean
306: Function CanSave( const Ext : string) : Boolean
307: Function CanStart( cChar : char) : boolean
308: Function CaptureScreen : TBitmap;
309: Function CaptureScreen1( Rec : TRect) : TBitmap;
310: Function CardinalToFourChar( ACARDINAL : LongWord) : string
311: Function CastSoapToNative(Info:PTypeInfo;const SoapData:WideString;NatData:Pointer;IsNull:Boolean): Boolean
312: Function CastSoapToVariant1( SoapInfo : PTypeInfo; const SoapData : WideString) : Variant;
313: Function Ceil( const X : Extended) : Integer
314: Function Ceil16( X : Integer) : Integer
315: Function Ceil4( X : Integer) : Integer
316: Function Ceil8( X : Integer) : Integer
317: Function Ceiling( const X : Extended) : Integer
318: Function CellRect( ACol, ARow : Longint) : TRect
319: Function CelsiusToFahrenheit( const AValue : Double) : Double
320: Function CenterPoint( const Rect : TRect) : TPoint
321: function CenterPoint(const Rect: TRect): TPoint)
322: Function ChangeFileExt( FileName, Extension : string) : string
323: function ChangeFileExt(const FileName: string; const Extension: string): string)
324: Function CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) : Boolean
325: Function CharInSet( const Ch : Char; const testSet: TSysCharSet): Boolean
326: Function CharIsEOF( const AString : string; ACharPos : Integer) : Boolean
327: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : String) : Boolean
328: Function CharLength( S : String; Index : Integer) : Integer
329: Function CharRange( const AMin, AMax : Char) : String
330: function CharsetToIdent(Charset: Longint; var Ident: string): Boolean)
331: Function CharToBin(vchr: Char): String;
332: Function CharNext(lpsz: PChar): PChar; stdcall;
333: Function CharToByteIndex( S : string; Index : Integer) : Integer
334: function CharToByteIndex(const S: string; Index: Integer): Integer)
335: Function CharToByteLen( S : string; MaxLen : Integer) : Integer
336: function CharToByteLen(const S: string; MaxLen: Integer): Integer)
337: Function CharToHex(const APrefix : String; const cc : Char) : shortstring;
338: function CharToHexStr(Value: char): string);
339: function CharToOem(ins, outs: PChar):boolean;
340: function CharToUniCode(Value: Char): string;
341: Function CheckMenuDropdown : Boolean
342: Function CheckMessages : longint
343: Function CheckBox: string;
344: Function CheckOpen( Status : DBIResult) : Boolean
345: Function CheckPassword( const APassword : String) : Boolean
346: Function CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt): SmallInt
347: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
348: function CheckSynchronize(Timeout: Integer): Boolean
349: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
350: Function CheckCom(AComNumber: Integer): Integer;');
351: Function CheckLPT1: string;');
352: function ChrA(const a: byte): char;
353: Function ClassIDToProgID(const ClassID: TGUID): string;
354: Function ClassNameIs(const Name: string): Boolean
355: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
356: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

789: function ExtractRelativePath(const BaseName: string; const DestName: string): string
790: Function ExtractShortPathName(FileName : string) : string
791: function ExtractShortPathName(const FileName: string): string
792: function ExtractStrings(Separators,WhiteSpace: TSysCharSet; Content: PChar;Strings: TStrings): Integer
793: function ExtractStrings(Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings): Integer
794: Function Fact(numb: integer): Extended;
795: Function FactInt(numb: integer): int64;
796: Function Factorial( const N: Integer ) : Extended
797: Function FahrenheitToCelsius( const AValue : Double ) : Double
798: function FalseBoolStrs: array of string
799: Function Fetch(var AInput:string;const ADelim:string;const ADelete:Bool;const ACaseSensitive:Bool):string
800: Function FetchCaseInsensitive(var AInput:string; const ADelim:string; const ADelete:Boolean): string
801: Function Fibo(numb: integer): Extended;
802: Function Fiboint(numb: integer): Int64;
803: Function Fibonacci( const N : Integer ) : Integer
804: Function FIELDBYNAME( const FIELDNAME : STRING ) : TFIELD
805: Function FIELDBYNAME( const FIELDNAME : WIDESTRING ) : TFIELD
806: Function FIELDBYNAME( const NAME : String ) : TFIELD
807: Function FIELDBYNAME( const NAME : String ) : TFIELDDEF
808: Function FIELDBYNUMBER( FIELDNO : INTEGER ) : TFIELD
809: Function FileAge( FileName : string ) : Integer
810: Function FileAge(const FileName: string): integer)
811: Function FileCompareText( const A, B : String ) : Integer
812: Function FileContains( const FileName:string;Text:string;CaseSensitive:Bool;ExceptionOnError:Bool): Boolean
813: Function FileCreate( FileName : string ) : Integer;//FileCreate2(FileName:string;Rights:Integer):Integer;
814: Function FileCreate(const FileName: string): integer)
815: Function FileCreateTemp( var Prefix : string ) : THandle
816: Function FileDateToDate( FileDate : Integer ) : TDateTime
817: function FileDateToDate( FileDate: Integer): TDateTime;
818: Function FileExists( const FileName : string ) : Boolean
819: Function FileExists( FileName : string ) : Boolean
820: function fileExists(const FileName: string): Boolean;
821: Function FileGetAttr( FileName : string ) : Integer
822: Function FileGetAttr(const FileName: string): integer)
823: Function FileGetDate( Handle : Integer ) : Integer
824: Function FileGetDate(handle: integer): integer
825: Function FileGetDisplayName( const FileName : string ) : string
826: Function FileGetSize( const FileName : string ) : Integer
827: Function FileGetTempName( const Prefix : string ) : string
828: Function FileGetType( const FileName : string ) : string
829: Function FileIsReadOnly( FileName : string ) : Boolean
830: Function FileLoad( ResType : TResType; const Name : string; MaskColor : TColor ) : Boolean
831: Function FileOpen( FileName : string; Mode : LongWord ) : Integer
832: Function FileOpen(const FileName: string; mode:integer): integer)
833: Function FileRead(handle: integer; Buffer: PChar; count: LongWord): integer
834: Function FileSearch( Name, DirList : string ) : string
835: Function FileSearch(const Name, dirList: string): string)
836: Function FileSeek( Handle : Integer; Offset : Int64; Origin : Integer ) : Int64;
837: Function FileSeek( Handle, Offset, Origin : Integer ) : Integer;
838: Function FileSeek(handle, offset, origin: integer): integer
839: Function FileSetAttr( FileName : string; Attr : Integer ) : Integer
840: function FileSetAttr(const FileName: string; Attr: Integer): Integer)
841: Function FileSetDate(FileName: string; Age : Integer) : Integer;
842: Function FileSetDate(handle: integer; age: integer): integer
843: Function FileSetDate2(FileHandle : Integer; Age : Integer) : Integer;
844: Function FileSetDateH( Handle : Integer; Age : Integer ) : Integer;
845: Function FileSetReadOnly( FileName : string; ReadOnly : Boolean ) : Boolean
846: Function FileSize( const FileName : string ) : int64
847: Function FileSizeByName( const AFilename : string ) : Longint
848: function FileWrite(Handle: Integer; const Buffer: pChar; Count: LongWord): Integer)
849: Function FilterSpecArray : TComdlgFilterSpecArray
850: Function FIND( ACAPTION : String ) : TMENUITEM
851: Function Find( AItem : Pointer; out AData : Pointer ) : Boolean
852: Function FIND( const ANAME : String ) : TNAMEDITEM
853: Function Find( const DisplayName : string ) : TAggregate
854: Function Find( const Item : TBookmarkStr; var Index : Integer ) : Boolean
855: Function FIND( const NAME : String ) : TFIELD
856: Function FIND( const NAME : String ) : TFIELDDEF
857: Function FIND( const NAME : String ) : TINDEXDEF
858: Function Find( const S : WideString; var Index : Integer ) : Boolean
859: function Find(S:String;var Index:Integer):Boolean
860: Function FindAuthClass( AuthName : String ) : TIAuthenticationClass
861: Function FindBand( AControl : TControl ) : TCoolBand
862: Function FindBoundary( AContentType : string ) : string
863: Function FindButton( AModalResult : TModalResult ) : TTaskDialogBaseButtonItem
864: Function FindCaption(StartIndex: Integer;Value: string; Partial,Inclusive,Wrap: Boolean): TListItem
865: Function FindCdLineSwitch( Switch : string; IgnoreCase : Boolean ) : Boolean;
866: Function FindCloseW(FindFile: Integer): LongBool; stdcall;
867: Function FindCmdLineSwitch( Switch : string; Chars : TSysCharSet; IgnoreCase : Boolean ) : Boolean;
868: Function FindCmmdLineSwitch( Switch : string ) : Boolean;
869: function FindComponent(AName: String): TComponent;
870: function FindComponent(vlabel: string): TComponent;
871: function FindComponent2(vlabel: string): TComponent;
872: function FindControl(Handle: HWnd): TWinControl;
873: Function FindData( StartIndex : Integer; Value : Pointer; Inclusive, Wrap : Boolean ) : TListItem
874: Function FindDatabase( const DatabaseName : string ) : TDatabase
875: function FindDragTarget(const Pos: TPoint; AllowDisabled: Boolean): TControl;
876: Function FINDFIELD( const FIELDNAME : STRING ) : TFIELD
877: Function FINDFIELD( const FIELDNAME : WideString ) : TFIELD

```

```

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

```

```

963: function FormatByteSize(const bytes: int64): string;
964: function FormatBuf(var Buffer:PChar;Buflen:Card;const Format:string;FmtLen:Card;const Args:array of const):Cardinal;
965: Function FormatCount( iCount : integer; const sSingular : string; const sPlural : string ) : string;
966: Function FormatCurr( Format : string; Value : Currency ) : string;
967: function FormatCurr(const Format: string; Value: Currency): string;
968: Function FormatDateTime( Format : string; DateTime : TDateTime ) : string;
969: function FormatDateTime(const fmt: string; D: TDateTime): string;
970: Function FormatFloat( Format : string; Value : Extended ) : string;
971: function FormatFloat(const Format: string; Value: Extended): string;
972: Function FormatFloat( Format : string; Value : Extended ) : string;
973: Function FormatFloat2( Format : string; Value : Extended; FormatSettings : TFormatSettings ) : string;
974: Function FormatCurr( Format : string; Value : Currency ) : string;
975: Function FormatCurr2(Format: string; Value: Currency; FormatSettings: TFormatSettings) : string;
976: Function Format2(const Format:string;const Args:array of const;const FSettings:TFormatSettings): string;
977: FUNCTION FormatInt(i: integer): STRING;
978: FUNCTION FormatInt64(i: int64): STRING;
979: Function FormatMaskText( const EditMask : string; const Value : string ) : string;
980: Function FormatValue( AValue : Cardinal ) : string;
981: Function FormatVersionString( const HiV, LoV : Word ) : string;
982: Function FormatVersionString1( const Major, Minor, Build, Revision : Word ) : string;
983: function Frac(X: Extended): Extended;
984: Function FreeResource( ResData : HGLOBAL ) : LongBool;
985: Function FromCommon( const AValue : Double ) : Double;
986: function FromCommon(const AValue: Double): Double;
987: Function FTPGMTDateToMLS( const ATimestamp : TDateTime; const AIncludeMsecs : Boolean ) : String;
988: Function FTPLocalDateToMLS( const ATimestamp : TDateTime; const AIncludeMsecs : Boolean ) : String;
989: Function FTPMLSToGMTDate( const ATimestamp : String ) : TDateTime;
990: Function FTPMLSToLocalDate( const ATimestamp : String ) : TDateTime;
991: Function FuncIn(AValue: Variant; ASet: Variant): Boolean;
992: //Function Funclist Size is: 6444 of mx3.9.8.9
993: Function FutureValue(const Rate:Extended;NPeriods:Integer;const Payment,PresentValue:Extended;PaymentTime:TPaymentTime):Extended;
994: Function FullTimeToStr(SUMTime: TDateTime): string;';
995: Function Gauss( const x, Spread : Double ) : Double;
996: function Gauss(const x,Spread: Double): Double;
997: Function GCD(x, y : LongInt) : LongInt;
998: Function GCDJ( X, Y : Cardinal ) : Cardinal;
999: Function GDAL: LongWord;
1000: Function GdiFlush : BOOL;
1001: Function GdiSetBatchLimit( Limit : DWORD ) : DWORD;
1002: Function GdiGetBatchLimit : DWORD;
1003: Function GenerateHeader : TIHeaderList;
1004: Function GeometricMean( const X : TDynFloatArray ) : Float;
1005: Function Get( AURL : string ) : string;
1006: Function Get2( AURL : string ) : string;
1007: Function Get8087CW : Word;
1008: function GetActiveOleObject(const ClassName: String): IDispatch;
1009: Function GetAliasDriverName( const AliasName : string ) : string;
1010: Function GetAPMBatteryFlag : TAPMBatteryFlag;
1011: Function GetAPMBatteryFullLifeTime : DWORD;
1012: Function GetAPMBatteryLifePercent : Integer;
1013: Function GetAPMBatteryLifeTime : DWORD;
1014: Function GetAPMLineStatus : TAPMLineStatus;
1015: Function GetAppdataFolder : string;
1016: Function GetAppDispatcher : TComponent;
1017: function GetArrayLength: integer;
1018: Function GetASCII: string;
1019: Function GetASCIILine: string;
1020: Function GetAsHandle( Format : Word ) : THandle;
1021: function GetAssociatedProgram(const Extension: string; var Filename, Description: string): boolean;
1022: Function GetBackupFileName( const FileName : string ) : string;
1023: function GetBaseAddress(PID:DWORD):DWORD; //Process API;
1024: Function GetBBitmap( Value : TBitmap ) : TBitmap;
1025: Function GetBIOSCopyright : string;
1026: Function GetBIOSDate : TDateTime;
1027: Function GetBIOSExtendedInfo : string;
1028: Function GetBIOSName : string;
1029: Function getBitmap(apath: string): TBitmap;
1030: Function GetBitmap( Index : Integer; Image : TBitmap ) : Boolean; //object;
1031: Function getBitmapObject(const bitmappath: string): TBitmap;
1032: Function GetButtonState( Button : TPageScrollerButton ) : TPageScrollerButtonState;
1033: Function GetCapsLockKeyState : Boolean;
1034: function GetCaptureControl: TControl;
1035: Function GetCDAudioTrackList( var TrackList : TJclCdTrackInfoArray; Drive : Char ) : TJclCdTrackInfo;
1036: Function GetCDAudioTrackList1( TrackList : TStrings; IncludeTrackType : Boolean; Drive : Char ) : string;
1037: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char ) : string;
1038: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char ) : string;
1039: Function GetClientThread( ClientSocket : TServerClientWinSocket ) : TServerClientThread;
1040: Function GetClockValue : Int64;
1041: function getCmdLine: PChar;
1042: function getCmdShow: Integer;
1043: function GetCPUSpeed: Double;
1044: Function GetColField( DataCol : Integer ) : TField;
1045: Function GetColorBlue( const Color : TColor ) : Byte;
1046: Function GetColorFlag( const Color : TColor ) : Byte;
1047: Function GetColorGreen( const Color : TColor ) : Byte;
1048: Function GetColorRed( const Color : TColor ) : Byte;
1049: Function GetComCtlVersion : Integer;

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

1489: Function IBFormatIdentifierValue( Dialect : Integer; Value : String ) : String
1490: Function IBExtractIdentifier( Dialect : Integer; Value : String ) : String
1491: Function IBQuoteIdentifier( Dialect : Integer; Value : String ) : String
1492: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1493: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1494: Function RandomString( iLength : Integer ) : String';
1495: Function RandomInteger( iLow, iHigh : Integer ) : Integer';
1496: Function StripString( st : String; CharsToStrip : String ) : String';
1497: FUNCTION Strip( const SubString: String; MainString: String): String;
1498: function StripTags(const S: string): string; //<'> of HTML
1499: function SizeToString(size : Int64; const unitStr : String) : String;
1500: FUNCTION NumberToString(No: Word): String;
1501: Function FormatIdentifier( Dialect : Integer; Value : String ) : String';
1502: Function FormatIdentifierValue( Dialect : Integer; Value : String ) : String';
1503: Function ExtractIdentifier( Dialect : Integer; Value : String ) : String';
1504: Function QuoteIdentifier( Dialect : Integer; Value : String ) : String';
1505: Function AddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1506: Procedure DecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String);
1507: function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;
1508: Function IconToBitmap( Ico : HICON ) : TBitmap
1509: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor ) : TBitmap
1510: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor ) : TBitmap
1511: function IdentToCharset(const Ident: string; var Charset: Longint): Boolean
1512: function IdentToColor(const Ident: string; var Color: Longint): Boolean
1513: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1514: Function IdGetDefaultCharSet : TIdCharSet
1515: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1516: Function IdPorts2 : TStringList
1517: Function IdToMib( const Id : string ) : string
1518: Function IdSHA1Hash(apath: string): string;
1519: Function IdHashSHA1(apath: string): string;
1520: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string ) : string
1521: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string ) : string;
1522: Function IfThenInt( AValue : Boolean; const ATrue : integer; AFalse : integer): integer';
1523: Function IfThenDouble( AValue : Boolean; const ATrue : double; AFalse : double): double';
1524: Function IfThenBool( AValue : Boolean; const ATrue : boolean; AFalse : boolean): boolean';
1525: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer ) : Integer;
1526: Function iif2( ATTest : Boolean; const ATrue : string; const AFalse : string ) : string;
1527: Function iif3( ATTest : Boolean; const ATrue : Boolean; const AFalse : Boolean ) : Boolean;
1528: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;
1529: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer ) : TDateTime
1530: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64 ) : TDateTime
1531: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte ) : Byte;
1532: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint ) : Shortint;
1533: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint ) : Smallint;
1534: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word ) : Word;
1535: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer ) : Integer;
1536: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal ) : Cardinal;
1537: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64 ) : Int64;
1538: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte ) : Byte;
1539: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint ) : Shortint;
1540: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint ) : Smallint;
1541: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word ) : Word;
1542: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer ) : Integer;
1543: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal ) : Cardinal;
1544: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64 ) : Int64;
1545: Function IncludeTrailingBackslash( S : string ) : string
1546: function IncludeTrailingBackslash(const S: string): string
1547: Function IncludeTrailingPathDelimiter( const APPath : string ) : string
1548: Function IncludeTrailingPathDelimiter( S : string ) : string
1549: function IncludeTrailingPathDelimiter(const S: string): string
1550: Function IncludeTrailingSlash( const APPath : string ) : string
1551: Function IncMillisecond( const AValue : TDateTime; const ANumberOfMilliseconds : Int64 ) : TDateTime
1552: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64 ) : TDateTime
1553: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer ) : TDateTime
1554: function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime
1555: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64 ) : TDateTime
1556: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer ) : TDateTime
1557: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer ) : TDateTime
1558: Function IndexOf( AClass : TClass ) : Integer
1559: Function IndexOf( AComponent : TComponent ) : Integer
1560: Function IndexOf( AObject : TObject ) : Integer
1561: Function INDEXOF( const ANAME : String ) : INTEGER
1562: Function IndexOf( const DisplayName : string ) : Integer
1563: Function IndexOf( const Item : TBookmarkStr ) : Integer
1564: Function IndexOf( const S : WideString ) : Integer
1565: Function IndexOf( const View : TJclFileMappingView ) : Integer
1566: Function INDEXOF( FIELD : TFIELD ) : INTEGER
1567: Function IndexOf( ID : LCID ) : Integer
1568: Function INDEXOF( ITEM : TMENUITEM ) : INTEGER
1569: Function IndexOf( Value : TListItem ) : Integer
1570: Function IndexOf( Value : TTreeNode ) : Integer
1571: function IndexOf(const S: string): Integer;
1572: Function IndexOfName( const Name : WideString ) : Integer
1573: function IndexOfName(Name: string): Integer;
1574: Function IndexOfObject( AObject : TObject ) : Integer
1575: function IndexOfObject(AObject:tObject):Integer
1576: Function IndexOfTabAt( X, Y : Integer ) : Integer
1577: Function IndexStr( const AText : string; const AValues : array of string ) : Integer

```

```

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

```

```

1665: Function ISEmpty : BOOLEAN
1666: Function IsEqual( Value : TParameters ) : Boolean
1667: Function ISEqual( VALUE : TPARAMS ) : BOOLEAN
1668: function IsEqualGUID( const guid1, guid2: TGUID): Boolean
1669: Function IsFirstNode : Boolean
1670: Function IsFloatZero( const X : Float ) : Boolean
1671: Function IsFormatRegistered( Extension, AppID : string ) : Boolean
1672: Function IsFormOpen( const FormName: string): Boolean;
1673: Function IsFQDN( const S : String ) : Boolean
1674: Function IsGrayScale : Boolean
1675: Function IsHex( AChar : Char ) : Boolean;
1676: Function IsHexString( const AString: string): Boolean;
1677: Function IsHostname( const S : String ) : Boolean
1678: Function IsInfinite( const AValue : Double) : Boolean
1679: Function IsInLeapYear( const AValue : TDateTime ) : Boolean
1680: Function IsInternet: boolean;
1681: Function IsLeadChar( ACh : Char ) : Boolean
1682: Function IsLeapYear( Year : Word) : Boolean
1683: function IsLeapYear(Year: Word): Boolean)
1684: function IsLibrary: Boolean)
1685: Function ISLINE : BOOLEAN
1686: Function IsLinkedTo( DataSet : TDataSet ) : Boolean
1687: Function ISLINKEDTO( DATASOURCE : TDATASOURCE ) : BOOLEAN
1688: Function IsLiteralChar( const EditMask : string; Offset : Integer ) : Boolean
1689: Function IsMatch( const Pattern, Text : string) : Boolean //Grep like RegEx
1690: Function IsMainAppWindow( Wnd : HWND ) : Boolean
1691: Function IsMediaPresentInDrive( Drive : Char ) : Boolean
1692: function IsMemoryManagerSet: Boolean)
1693: Function IsMultiTableQuery( const SQL : WideString ) : Boolean
1694: function IsMultiThread: Boolean)
1695: Function IsNumeric( AChar : Char ) : Boolean;
1696: Function IsNumeric2( const AString : string ) : Boolean;
1697: Function IsNTFS: Boolean;
1698: Function IsOctal( AChar : Char ) : Boolean;
1699: Function IsOctalString(const AString: string) : Boolean;
1700: Function IsPathDelimiter( S : string; Index : Integer ) : Boolean
1701: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1702: Function IsPM( const AValue : TDateTime ) : Boolean
1703: Function IsPositiveFloatArray( const X : TDynFloatArray ) : Boolean
1704: Function IsPortAvailable( ComNum : Cardinal ) : Boolean');
1705: Function IsComPortReal( ComNum : Cardinal ) : Boolean');
1706: Function IsCOM( ComNum : Cardinal ) : Boolean');
1707: Function IsComPort: Boolean');
1708: Function IsPrimeFactor( const F, N : Cardinal ) : Boolean
1709: Function IsPrimerM( N : Cardinal ) : Boolean //rabin miller
1710: Function IsPrimeTD( N : Cardinal ) : Boolean //trial division
1711: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1712: Function ISqr( const I : Smallint ) : Smallint
1713: Function IsReadOnly(const Filename: string): boolean;
1714: Function IsRectEmpty( const Rect : TRect ) : Boolean
1715: function IsRectEmpty(const Rect): Boolean)
1716: Function IsRelativePrime( const X, Y : Cardinal ) : Boolean
1717: Function ISRIGHTTOLEFT : BOOLEAN
1718: function IsRightToLeft: Boolean
1719: Function IsSameDay( const AValue, ABasis : TDateTime ) : Boolean
1720: Function ISSEQUENCED : BOOLEAN
1721: Function IsSystemModule( const Module : HMODULE ) : Boolean
1722: Function IsSystemResourcesMeterPresent : Boolean
1723: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1724: Function IsToday( const AValue : TDateTime ) : Boolean
1725: function IsToday(const AValue: TDateTime): Boolean;
1726: Function IsTopDomain( const AStr : string ) : Boolean
1727: Function IsUTF8LeadByte( Lead : Char ) : Boolean
1728: Function IsUTF8String( const s : UTF8String ) : Boolean
1729: Function IsUTF8TrailByte( Lead : Char ) : Boolean
1730: Function ISVALIDCHAR( INPUTCHAR : CHAR ) : BOOLEAN
1731: Function IsValidDate( const AYear, AMonth, ADay : Word ) : Boolean
1732: Function IsValidDateDay( const AYear, ADayOfYear : Word ) : Boolean
1733: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word ) : Boolean
1734: Function IsValidDateTime( const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1735: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word ) : Boolean
1736: Function IsValidIdent( Ident : string ) : Boolean
1737: function IsValidIdent1(const Ident: string; AllowDots: Boolean): Boolean)
1738: Function IsValidIP( const S : String ) : Boolean
1739: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word ) : Boolean
1740: Function IsValidPNG(stream: TStream): Boolean;
1741: Function IsValidJPEG(stream: TStream): Boolean;
1742: Function IsValidISBN( const ISBN : AnsiString ) : Boolean
1743: Function IsVariantManagerSet: Boolean; //deprecated;
1744: Function IsVirtualPcGuest : Boolean;
1745: Function IsVmWareGuest : Boolean;
1746: Function IsVCLControl(Handle: HWnd): Boolean;
1747: Function IsWhiteString( const AStr : String ) : Boolean
1748: Function IsWindowResponding( Wnd : HWND; Timeout : Integer ) : Boolean
1749: Function IsWo64: boolean;
1750: Function IsWin64: boolean;
1751: Function IsWow64String(var s: string): Boolean;
1752: Function IsWin64String(var s: string): Boolean;
1753: Function IsWindowsVista: boolean;

```

```

1754: Function isPowerof2(num: int64): boolean;
1755: Function powerOf2(exponent: integer): int64;
1756: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1757: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1758: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1759: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1760: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1761: Function ItemRect( Index : Integer) : TRect
1762: function ITEMRECT(INDEX:INTEGER):TRECT
1763: Function ItemWidth( Index : Integer) : Integer
1764: Function JavahashCode(val: string): Integer;
1765: Function JosephusG(n,k: integer; var graphout: string): integer;
1766: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1767: Function JustName(PathName: string) : string; //in path and ext
1768: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1769: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1770: Function KeepAlive : Boolean
1771: Function KeysToShiftState(Keys: Word): TShiftState;
1772: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1773: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1774: Function KeyboardStateToShiftState: TShiftState; overload;
1775: Function Languages : TLanguages
1776: Function Last : TClass
1777: Function Last : TComponent
1778: Function Last : TObject
1779: Function LastDelimiter( Delimiters, S : string) : Integer
1780: function LastDelimiter(const Delimiters: string; const S: string): Integer
1781: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1782: Function Latitude2WGS84(lat: double): double;
1783: Function LCM(m,n:longint):longint;
1784: Function LCMJ( const X, Y : Cardinal) : Cardinal
1785: Function Ldexp( const X : Extended; const P : Integer) : Extended
1786: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1787: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1788: function Length: Integer;
1789: Procedure LetFileList(FileList: TStringlist; apath: string);
1790: function lengthmp3(mp3path: string):integer;
1791: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1792: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1793: function LinesCount(sfilename:string):extended;
1794: function TextfileLineCount(const FileName: string): integer;
1795: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
  L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1796: function LineStart(Buffer, BufPos: PChar): PChar
1797: function LineStart(Buffer, BufPos: PChar): PChar
1798: function ListSeparator: char;
1799: function Ln(x: Extended): Extended;
1800: Function LnXP1( const X : Extended) : Extended
1801: function Lo(vdat: word): byte;
1802: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1803: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1804: Function LoadFileAsString( const FileName : string) : string
1805: Function LoadFromFile( const FileName : string) : TBitmapLoader
1806: function Loadfile(const FileName: TFileName): string;
1807: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1808: Function LoadPackage(const Name: string): HMODULE
1809: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1810: Function LoadStr( Ident : Integer) : string
1811: Function LoadString(Instance: Longint; Ident: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1812: Function LoadWideStr( Ident : Integer) : WideString
1813: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1814: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1815: Function LockServer( fLock : LongBool) : HResult
1816: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1817: Function Log( const X : Extended) : Extended
1818: Function Log10( const X : Extended) : Extended
1819: Function Log2( const X : Extended) : Extended
1820: function LogBase10(X: Float): Float;
1821: Function LogBase2(X: Float): Float;
1822: Function LogBaseN(Base, X: Float): Float;
1823: Function LogN( const Base, X : Extended) : Extended
1824: Function LogOffOS : Boolean
1825: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1826: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1827: Function LongDateFormat: string;
1828: function LongTimeFormat: string;
1829: Function LongWordToFourChar( ACardinal : LongWord) : string
1830: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1831: Function LookupName( const name : string) : TInAddr
1832: Function LookupService( const service : string) : Integer
1833: function Low: Int64;
1834: Function LowerCase( S : string) : string
1835: Function Lowercase(s : AnyString) : AnyString;
1836: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1837: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1838: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1839: function MainInstance: longword
1840: function MainThreadID: longword
1841: Function Map(x, in_min, in_max, out_min, out_max: integer): integer; //arduino

```

```

1842: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1843: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1844: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1845: Function MakeIDB( out Bitmap : PBitmapInfo) : Integer
1846: Function MakeWordIntoIPv4Address( const ADWord : Cardinal) : string
1847: Function Makefile(const FileName: string): integer)';
1848: function MakeLong(A, B: Word): Longint)
1849: Function MakeTempFilename( const APPath : String) : string
1850: Function MakeValidFileName( const Str : string) : string
1851: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1852: function MakeWord(A, B: Byte): Word)
1853: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1854: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1855: Function MapValues( Mapping : string; Value : string) : string
1856: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1857: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1858: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1859: Function MaskGetFldSeparator( const EditMask : string) : Integer
1860: Function MaskGetMaskBlank( const EditMask : string) : Char
1861: Function MaskGetMaskSave( const EditMask : string) : Boolean
1862: Function MaskIntLiteralToChar( IChar : Char) : Char
1863: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1864: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1865: Function MaskString( Mask, Value : String) : String
1866: Function Match( const sString : string) : TniRegularExpressionMatchResul
1867: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1868: Function Matches( const Filename : string) : Boolean
1869: Function MatchesMask( const Filename, Mask : string) : Boolean
1870: Function MatchStr( const ATText : string; const AValues : array of string) : Boolean
1871: Function MatchText( const ATText : string; const AValues : array of string) : Boolean
1872: Function Max( AValueOne, AValueTwo : Integer) : Integer
1873: function Max(const x,y: Integer): Integer;
1874: Function Max1( const B1, B2 : Shortint) : Shortint;
1875: Function Max2( const B1, B2 : Smallint) : Smallint;
1876: Function Max3( const B1, B2 : Word) : Word;
1877: function Max3(const x,y,z: Integer): Integer;
1878: Function Max4( const B1, B2 : Integer) : Integer;
1879: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1880: Function Max6( const B1, B2 : Int64) : Int64;
1881: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1882: Function MaxFloat( const X, Y : Float) : Float
1883: Function MaxFloatArray( const B : TDynFloatArray) : Float
1884: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1885: function MaxIntValue(const Data: array of Integer):Integer)
1886: Function MaxJ( const B1, B2 : Byte) : Byte;
1887: function MaxPath: string;
1888: function MaxValue(const Data: array of Double): Double)
1889: Function MaxCalc( const Formula : string) : Extended //math expression parser
1890: Procedure MaxCalcF( const Formula : string); //out to console memo2
1891: Function MaxCalcs( const Formula : string): string)';
1892: function MD5(const fileName: string): string;
1893: Function Mean( const Data : array of Double) : Extended
1894: Function Median( const X : TDynFloatArray) : Float
1895: Function Memory : Pointer
1896: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1897: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1898: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1899: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1900: Function MessageDlg1l(const
  Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1901: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
  Y:Integer):Integer;
1902: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
  Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn ) : Integer;
1903: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
  Longint; X, Y : Integer; const HelpFileName : string ) : Integer;
1904: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
  Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn ) : Integer;
1905: Function MibToId( Mib : string) : string
1906: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1907: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1908: Function microsecondsToCentimeters(milliseconds: longint): longint; //340m/s speed of sound
1909: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1910: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1911: Procedure GetMidiOutputs( const List : TStrings)
1912: // GetGEOMAPX('html',ExePath+cologne2mapX.html','cathedral cologne')
1913: Procedure GetGEOMap(C_form,apath: string; const Data: string); //C_form: [html/json/xml]
1914: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1915: Function MIDINoteToStr( Note : TMIDINote) : string
1916: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1917: Procedure GetMidiOutputs( const List : TStrings)
1918: Procedure MidiOutCheck( Code : MMResult)
1919: Procedure MidiInCheck( Code : MMResult)
1920: Function MillisecondOf( const AValue : TDateTime) : Word
1921: Function MillisecondOfTheDay( const AValue : TDateTime) : LongWord
1922: Function MillisecondOfTheHour( const AValue : TDateTime) : LongWord
1923: Function MillisecondOfTheMinute( const AValue : TDateTime) : LongWord
1924: Function MillisecondOfTheMonth( const AValue : TDateTime) : LongWord
1925: Function MillisecondOfTheSecond( const AValue : TDateTime) : Word

```

```

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

```

```

2010: Function Now : TDateTime
2011: function Now2: tDateTime
2012: Function NumProcessThreads : integer
2013: Function NumThreadCount : integer
2014: Function NthDayOfWeek( const AValue : TDateTime) : Word
2015: Function NtProductType : TNTProductType
2016: Function NtProductTypeString : string
2017: function Null: Variant;
2018: Function NullPoint : TPoint
2019: Function NullRect : TRect
2020: Function Null2Blank(aString:String):String;
2021: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
Extended; PaymentTime : TPaymentTime) : Extended
2022: Function NumIP : integer
2023: function Odd(x: Longint): boolean;
2024: Function OffsetFromUTC : TDateTime
2025: Function OffsetPoint( const P, Offset : TPoint) : TPoint
2026: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
2027: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
2028: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
2029: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
2030: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
2031: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
2032: function OpenBit:Integer
2033: Function OpenDatabase : TDatabase
2034: Function OpenDatabase( const DatabaseName : string) : TDatabase
2035: Procedure OpenDir(adir: string);
2036: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
2037: Function OpenMap(const Data: string): boolean;
2038: Function OpenMapX(const Data: string): boolean;
2039: Function OpenObject( Value : PChar) : Boolean;
2040: Function OpenObject1( Value : string) : Boolean;
2041: Function OpenSession( const SessionName : string) : TSession
2042: Function OpenVolume( const Drive : Char) : THandle
2043: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
2044: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
2045: Function OrdToBinary( const Value : Byte) : string;
2046: Function OrdToBinary1( const Value : Shortint) : string;
2047: Function OrdToBinary2( const Value : Smallint) : string;
2048: Function OrdToBinary3( const Value : Word) : string;
2049: Function OrdToBinary4( const Value : Integer) : string;
2050: Function OrdToBinary5( const Value : Cardinal) : string;
2051: Function OrdToBinary6( const Value : Int64) : string;
2052: Function OSCheck( RetVal : Boolean) : Boolean
2053: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
2054: Function OSIdentToString( const OSIdent : DWORD) : string
2055: Function Output: Text)
2056: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
2057: Function Owner : TCustomListView
2058: function Owner : TPersistent
2059: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
2060: Function PadL( pStr : String; pLth : integer) : String
2061: Function Padl(s : AnyString;I : longInt) : AnyString;
2062: Function PadLCh( pStr : String; plth : integer; pChr : char) : String
2063: Function PadR( pStr : String; pLth : integer) : String
2064: Function Padr(s : AnyString;I : longInt) : AnyString;
2065: Function PadRCh( pStr : String; plth : integer; pChr : char) : String
2066: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2067: Function Padz(s : AnyString;I : longInt) : AnyString;
2068: Function PaethPredictor( a, b, c : Byte) : Byte
2069: Function PARAMBYNAME( const VALUE : String) : TPARAM
2070: Function ParamByName( const Value : WideString) : TParameter
2071: Function ParamCount: Integer
2072: Function ParamsEncode( const ASrc : string) : string
2073: function ParamStr(Index: Integer): string)
2074: Function ParseDate( const DateStr : string) : TDateTime
2075: Function PARSESQL( SQL : string; DOCREATE : BOOLEAN) : String
2076: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2077: Function PathAddExtension( const Path, Extension : string) : string
2078: Function PathAddSeparator( const Path : string) : string
2079: Function PathAppend( const Path, Append : string) : string
2080: Function PathBuildRoot( const Drive : Byte) : string
2081: Function PathCanonicalize( const Path : string) : string
2082: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2083: Function PathCompactPath(const DC: HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2084: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2085: Function PathEncode( const ASrc : string) : string
2086: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2087: Function PathExtractFileNameNoExt( const Path : string) : string
2088: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2089: Function PathGetDepth( const Path : string) : Integer
2090: Function PathGetLongName( const Path : string) : string
2091: Function PathGetLongName2( Path : string) : string
2092: Function PathGetShortName( const Path : string) : string
2093: Function PathIsAbsolute( const Path : string) : Boolean
2094: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2095: Function PathIsDiskDevice( const Path : string) : Boolean
2096: Function PathIsUNC( const Path : string) : Boolean
2097: Function PathRemoveExtension( const Path : string) : string

```

```

2098: Function PathRemoveSeparator( const Path : string ) : string
2099: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2100: Function Peek : Pointer
2101: Function Peek : TObject
2102: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2103: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
Extended; PaymentTime : TPaymentTime) : Extended
2104: function Permutation(npr, k: integer): extended;
2105: function PermutationInt(npr, k: integer): Int64;
2106: Function PermutationJ( N, R : Cardinal ) : Float
2107: Function Pi : Extended;
2108: Function PiE : Extended;
2109: Function PixelsToDialogsX( const Pixels : Word ) : Word
2110: Function PixelsToDialogsY( const Pixels : Word ) : Word
2111: Function PlaySound(s: pchar; flag, syncflag: integer): boolean;
2112: Function Point( X, Y : Integer ) : TPoint
2113: function Point(X, Y: Integer): TPoint
2114: Function PointAssign( const X, Y : Integer ) : TPoint
2115: Function PointDist( const P1, P2 : TPoint ) : Double;
2116: function PointDist(const P1,P2: TFloatPoint): Double;
2117: Function PointDist1( const P1, P2 : TFloatPoint ) : Double;
2118: function PointDist2(const P1,P2: TPoint): Double;
2119: Function PointEqual( const P1, P2 : TPoint ) : Boolean
2120: Function PointIsNull( const P : TPoint ) : Boolean
2121: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint ) : Double
2122: Function Poly( const X : Extended; const Coefficients : array of Double ) : Extended
2123: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2124: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2125: Function Pop : Pointer
2126: Function Pop : TObject
2127: Function PopnStdDev( const Data : array of Double ) : Extended
2128: Function PopnVariance( const Data : array of Double ) : Extended
2129: Function PopulationVariance( const X : TDynFloatArray ) : Float
2130: function Pos(SubStr, S: AnyString): Longint;
2131: Function PosEqual( const Rect : TRect ) : Boolean
2132: Function PosEx( const SubStr, S : string; Offset : Integer ) : Integer
2133: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt ) : Integer
2134: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2135: Function Post1( AURL : string; const ASource : TStrings ) : string;
2136: Function Post2( AURL : string; const ASource : TStream ) : string;
2137: Function Post3( AURL : string; const ASource : TIIdMultiPartFormDataStream ) : string;
2138: Function PostData( const UserData : WideString; const CheckSum : DWORD ) : Boolean
2139: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2140: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2141: Function Power( const Base, Exponent : Extended ) : Extended
2142: Function PowerBig(aval, n:integer): string;
2143: Function PowerIntJ( const X : Float; N : Integer ) : Float;
2144: Function PowerJ( const Base, Exponent : Float ) : Float;
2145: Function PowerOffOS : Boolean
2146: Function PreformatDateString( Ps : string ) : string
2147: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2148: Function PrimeFactors( N : Cardinal ) : TDynCardinalArray
2149: Function Printer : TPrinter
2150: Function ProcessPath2( const ABasePath:String; const APPath: String; const APathDelim:string): string
2151: Function ProcessResponse : TIIdHTTPWhatsNext
2152: Function ProduceContent : string
2153: Function ProduceContentFromStream( Stream : TStream ) : string
2154: Function ProduceContentFromString( const S : string ) : string
2155: Function ProgIDToClassID(const ProgID: string): TGUID;
2156: Function PromptDataLinkfile( ParentHandle : THandle; InitialFile : WideString ) : WideString
2157: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString ) : WideString
2158: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
const ATitle : string; const AInitialDir : string; SaveDialog : Boolean ) : Boolean
2159: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
ATitle: string; const AInitialDir: string; SaveDialog: Boolean)
2160: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2161: Function PtInRect( const Rect : TRect; const P : TPoint ) : Boolean
2162: function PtInRect(const Rect: TRect; const P: TPoint): Boolean
2163: Function Push( AItem : Pointer ) : Pointer
2164: Function Push( AObject : TObject ) : TObject
2165: Function Put1( AURL : string; const ASource : TStream ) : string;
2166: Function Pythagoras( const X, Y : Extended ) : Extended
2167: Function queryDLLInterface( var queryList : TStringList ) : TStringList
2168: Function queryDLLInterfaceTwo( var queryList : TStringList ) : TStringList
2169: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2170: Function queryPerformanceCounter2(mse: int64): int64;
2171: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2172: //Function QueryPerformanceFrequency(msc: int64): boolean;
2173: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2174: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2175: Procedure QueryPerformanceCounter1(var aC: Int64);
2176: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2177: Function Quote( const ACommand : String ) : SmallInt
2178: Function QuotedStr( S : string ) : string
2179: Function RadToCycle( const Radians : Extended ) : Extended
2180: Function RadToDeg( const Radians : Extended ) : Extended
2181: Function RadToDeg( const Value : Extended ) : Extended;

```

```

2182: Function RadToDeg1( const Value : Double) : Double;
2183: Function RadToDeg2( const Value : Single) : Single;
2184: Function RadToGrad( const Radians : Extended) : Extended;
2185: Function RadToGrad( const Value : Extended) : Extended;
2186: Function RadToGrad1( const Value : Double) : Double;
2187: Function RadToGrad2( const Value : Single) : Single;
2188: Function RandG( Mean, StdDev : Extended) : Extended;
2189: function Random(const ARange: Integer): Integer;
2190: function random2(a: integer): double;
2191: function RandomE: Extended;
2192: function RandomF: Extended;
2193: Function RandomFrom( const AValues : array of string) : string;
2194: Function RandomRange( const AFrom, ATo : Integer) : Integer;
2195: function randSeed: longint;
2196: Function RawToDataColumn( ACol : Integer) : Integer;
2197: Function Read : Char;
2198: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult;
2199: function Read(Buffer:String;Count:LongInt):LongInt;
2200: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer;
2201: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean;
2202: Function ReadCardinal( const AConvert : boolean) : Cardinal;
2203: Function ReadChar : Char;
2204: Function ReadClient( var Buffer, Count : Integer) : Integer;
2205: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime;
2206: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime;
2207: Function ReadFloat( const Section, Name : string; Default : Double) : Double;
2208: Function ReadFromStack(const ARaiseExceptfDiscn:Bool;ATimeout:Int;const ARaiseExceptTimeout:Bool):Int;
2209: Function ReadInteger( const AConvert : boolean) : Integer;
2210: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint;
2211: Function ReadLn : string;
2212: Function ReadLn(ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string;
2213: function ReadLn(question: string): string;
2214: Function ReadLn: string; //read last line in memo2 - console!
2215: Function ReadLnWait( AFailCount : Integer) : string;
2216: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2217: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2218: Function ReadSmallInt( const AConvert : boolean) : SmallInt;
2219: Function ReadString( const ABytes : Integer) : string;
2220: Function ReadString( const Section, Ident, Default : string) : string;
2221: Function ReadString( Count : Integer) : string;
2222: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime;
2223: Function ReadTimeStampCounter : Int64;
2224: Function RebootOS : Boolean;
2225: Function Receive( ATimeOut : Integer) : TReplyStatus;
2226: Function ReceiveBuf( var Buf, Count : Integer) : Integer;
2227: Function ReceiveLength : Integer;
2228: Function ReceiveText : string;
2229: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal;
2230: Function ReceiveSerialText: string;
2231: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime;
2232: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,AMilliSec:Word):TDateTime;
2233: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime;
2234: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime;
2235: Function RecodeMillisecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime;
2236: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime;
2237: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime;
2238: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime;
2239: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime;
2240: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime;
2241: Function Reconcile( const Results : OleVariant) : Boolean;
2242: Function Rect( Left, Top, Right, Bottom : Integer) : TRect;
2243: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect;
2244: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2245: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect;
2246: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect;
2247: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect;
2248: Function RectCenter( const R : TRect) : TPoint;
2249: Function RectEqual( const R1, R2 : TRect) : Boolean;
2250: Function RectHeight( const R : TRect) : Integer;
2251: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean;
2252: Function RectIncludesRect( const R1, R2 : TRect) : Boolean;
2253: Function RectIntersection( const R1, R2 : TRect) : TRect;
2254: Function RectIntersectRect( const R1, R2 : TRect) : Boolean;
2255: Function RectIsEmpty( const R : TRect) : Boolean;
2256: Function RectIsNull( const R : TRect) : Boolean;
2257: Function RectIsSquare( const R : TRect) : Boolean;
2258: Function RectIsValid( const R : TRect) : Boolean;
2259: Function RectsAreValid( R : array of TRect) : Boolean;
2260: Function RectUnion( const R1, R2 : TRect) : TRect;
2261: Function RectWidth( const R : TRect) : Integer;
2262: Function RedComponent( const Color32 : TColor32) : Integer;
2263: Function Refresh : Boolean;
2264: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2265: Function RegisterConversionFamily( const ADescription : string) : TConvFamily;
2266: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2267: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType;
2268: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2269: Function ReleaseDC(hwnd: HWND; hdc: HDC): integer;

```

```

2270: Function ReleaseHandle : HBITMAP
2271: Function ReleaseHandle : HENHMETAFILE
2272: Function ReleaseHandle : HICON
2273: Function ReleasePalette : HPALETTE
2274: Function RemainderFloat( const X, Y : Float) : Float
2275: Function Remove( AClass : TClass) : Integer
2276: Function Remove( AComponent : TComponent) : Integer
2277: Function Remove( AItem : Integer) : Integer
2278: Function Remove( AItem : Pointer) : Pointer
2279: Function Remove( AItem : TObject) : TObject
2280: Function Remove( AObject : TObject) : Integer
2281: Function RemoveBackslash( const PathName : string) : string
2282: Function RemoveDF( aString : String) : String //removes thousand separator
2283: Function RemoveDir( Dir : string) : Boolean
2284: function RemoveDir(const Dir: string): Boolean
2285: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2286: Function RemoveFileExt( const FileName : string) : string
2287: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2288: Function RenameFile( OldName, NewName : string) : Boolean
2289: function RenameFile(const OldName: string; const NewName: string): Boolean
2290: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2291: Function ReplaceText( const AText, AFromText, AToText : string) : string
2292: Function Replicate(c : char; i : longInt) : String;
2293: Function Request : TWebRequest
2294: Function ResemblesText( const AText, AOther : string) : Boolean
2295: Function Reset : Boolean
2296: function Reset2(mypath: string): TStringlist //string;
2297: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2298: Function ResourceLoad( Restype: TResType; const Name : string; MaskColor : TColor) : Boolean
2299: Function Response : TWebResponse
2300: Function ResumeSupported : Boolean
2301: Function RETHINKHOTKEYS : BOOLEAN
2302: Function RETHINKLINES : BOOLEAN
2303: Function Retrieve( const MsgNum : Integer; AMsg : TIIdMessage) : Boolean
2304: Function RetrieveCurrentDir : string
2305: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2306: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIIdMessage) : Boolean
2307: Function RetrieveMailBoxSize : integer
2308: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2309: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2310: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStringList) : boolean
2311: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2312: Function ReverseBits( Value : Byte) : Byte;
2313: Function ReverseBits1( Value : Shortint) : Shortint;
2314: Function ReverseBits2( Value : Smallint) : Smallint;
2315: Function ReverseBits3( Value : Word) : Word;
2316: Function ReverseBits4( Value : Cardinal) : Cardinal;
2317: Function ReverseBits4( Value : Integer) : Integer;
2318: Function ReverseBits5( Value : Int64) : Int64;
2319: Function ReverseBytes( Value : Word) : Word;
2320: Function ReverseBytes1( Value : Smallint) : Smallint;
2321: Function ReverseBytes2( Value : Integer) : Integer;
2322: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2323: Function ReverseBytes4( Value : Int64) : Int64;
2324: Function ReverseString( const AText : string) : string
2325: Function ReverseDNSLookup(const IPAddrs:String;DNSServer:String;Timeout:Int;var HName:Str):Bool;
2326: Function Revert : HRESULT
2327: Function RGB(R,G,B: Byte): TColor;
2328: Function RGB2BGR( const Color : TColor) : TColor
2329: Function RGB2TColor( R, G, B : Byte) : TColor
2330: Function RGBToWebColorName( RGB : Integer) : string
2331: Function RGBToWebColorStr( RGB : Integer) : string
2332: Function RgbToHtml( Value : TColor) : string
2333: Function HtmlToRgb(const Value: string): TColor;
2334: Function RightStr( const AStr : String; Len : Integer) : String
2335: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2336: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2337: Function ROL( Aval : LongWord; AShift : Byte) : LongWord
2338: Function ROR( Aval : LongWord; AShift : Byte) : LongWord
2339: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2340: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2341: Function Round(e : Extended) : Longint;
2342: Function Round64(e: extended): Int64;
2343: Function RoundAt( const Value : string; Position : SmallInt) : string
2344: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2345: Function RoundTo(const AValue: Extended; const ADigit: TRoundToEXRangeExtended): Extended;'+';
2346: Function SimpleRoundTo(const AValue: Extended; const Adigit: TRoundToRange): Extended;'+';
2347: Function RoundFrequency( const Frequency : Integer) : Integer
2348: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2349: Function RoundPoint( const X, Y : Double) : TPoint
2350: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2351: Function RowCount : Integer
2352: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2353: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2354: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2355: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2356: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2357: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2358: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean

```

```

2359: Function RunningProcessesList( const List : TStrings; FullPath : Boolean ) : Boolean
2360: Function RunByteCode(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;'
2361: Function RunCompiledScript2(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;'
2362: Function S_AddBackSlash( const ADirName : string ) : string
2363: Function S_AllTrim( const cStr : string ) : string
2364: Function S_AtRepl( const cAT, cStr, cRepl : string ) : string
2365: Function S_Cut( const cStr : string; const iLen : integer ) : string
2366: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer ) : integer
2367: Function S_DirExists( const ADir : string ) : Boolean
2368: Function S_Empty( const cStr : string ) : boolean
2369: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer ) : string
2370: Function S_LargeFontsActive : Boolean
2371: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer ) : Extended
2372: Function S_LTrim( const cStr : string ) : string
2373: Function S_ReadNextTextLineFromStream( stream : TStream ) : string
2374: Function S_RepeatChar( const iLen : integer; const AChar : Char ) : String
2375: Function S_ReplFirst( const cAT, cStr, cRepl : string ) : string
2376: Function S_RoundDecimal( AValue : Extended; APPlaces : Integer ) : Extended
2377: Function S_RTrim( const cStr : string ) : string
2378: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer ) : string
2379: //Type TS_ShellExecuteCmd = (seCmdOpen, seCmdPrint, seCmdExplore);
2380: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd ) : string
2381: Function S_Space( const iLen : integer ) : String
2382: Function S_StrBlanks( const cStr : string; const iLen : integer ) : string
2383: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer ) : string
2384: Function S_StrCRC32( const Text : string ) : LongWORD
2385: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string
2386: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string
2387: Function S_StringtoUTF_8( const AString : string ) : string
2388: Function S_StrLBlanks( const cStr : string; const iLen : integer ) : string
2389: function S_StrToReal(const cStr: string; var R: Double): Boolean
2390: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean ) : boolean
2391: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean ) : string
2392: Function S_UTF_8ToString( const AString : string ) : string
2393: Function S_WBox( const AText : string ) : integer
2394: Function SameDate( const A, B : TDateTime ) : Boolean
2395: function SameDate( const A, B : TDateTime ): Boolean;
2396: Function SameDateTime( const A, B : TDateTime ) : Boolean
2397: function SameDateTime( const A, B : TDateTime ): Boolean;
2398: Function SameFileName( S1, S2 : string ) : Boolean
2399: Function SameText( S1, S2 : string ) : Boolean
2400: function SameText( const S1: string; const S2: string): Boolean)
2401: Function SameTime( const A, B : TDateTime ) : Boolean
2402: function SameTime( const A, B : TDateTime ): Boolean;
2403: function SameValue( const A, B: Extended; Epsilon: Extended): Boolean //overload;
2404: function SameValue1( const A, B: Double; Epsilon: Double): Boolean //overload;
2405: function SameValue2( const A, B: Single; Epsilon: Single): Boolean //overload;
2406: Function SampleVariance( const X : TDynFloatArray ) : Float
2407: Function Sar( const Value : Shortint; const Count : TBitRange ) : Shortint;
2408: Function Sar1( const Value : Smallint; const Count : TBitRange ) : Smallint;
2409: Function Sar2( const Value : Integer; const Count : TBitRange ) : Integer;
2410: Function SaveToFile( const AFileName : TFileName ) : Boolean
2411: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2412: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2413: Function ScanF(const aformat: String; const args: array of const): string;
2414: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2415: Function SearchBuf(Buf:PChar; BufLen:Integer; SelStart,SelLength:Integer; searchString:String; Options: TStringSearchOptions):PChar
2416: Function SearchBuf2(Buf: String; SelStart, SelLength: Integer; searchString: String; Options: TStringSearchOptions): Integer;
2417: function SearchRecattr: integer;
2418: function SearchRecExcludeAttr: integer;
2419: Function SearchRecFileSize64( const SearchRec : TSearchRec ) : Int64
2420: function SearchRecname: string;
2421: function SearchRecsize: integer;
2422: function SearchRectime: integer;
2423: Function Sec( const X : Extended ) : Extended
2424: Function Secant( const X : Extended ) : Extended
2425: Function SecH( const X : Extended ) : Extended
2426: Function SecondOf( const AValue : TDateTime ) : Word
2427: Function SecondOfTheDay( const AValue : TDateTime ) : LongWord
2428: Function SecondOfTheHour( const AValue : TDateTime ) : Word
2429: Function SecondOfTheMinute( const AValue : TDateTime ) : Word
2430: Function SecondOfTheMonth( const AValue : TDateTime ) : LongWord
2431: Function SecondOfTheWeek( const AValue : TDateTime ) : LongWord
2432: Function SecondOfTheYear( const AValue : TDateTime ) : LongWord
2433: Function SecondsBetween( const ANow, AThen : TDateTime ) : Int64
2434: Function SecondSpan( const ANow, AThen : TDateTime ) : Double
2435: Function SectionExists( const Section : string ) : Boolean
2436: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption ) : Boolean
2437: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint ) : HResult
2438: function Seek(Offset:LongInt-Origin:Word):LongInt
2439: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2440: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string; Options : TSelectDirExtOpts; Parent : TwinControl ) : Boolean;
2441: Function SelectImage( var AFileName : string; const Extensions, Filter : string ) : Boolean
2442: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2443: Function SendBuf( var Buf, Count : Integer ) : Integer
2444: Function SendCmd( const AOut : string; const AResponse : SmallInt ) : SmallInt;

```

```

2445: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2446: Function SendKey( AppName : string; Key : Char) : Boolean
2447: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2448: Function SendStream( AStream : TStream) : Boolean
2449: Function SendStreamThenDrop( AStream : TStream) : Boolean
2450: Function SendText( const S : string) : Integer
2451: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2452: Function SendSerialText(Data: String): cardinal
2453: Function Sent : Boolean
2454: Function ServicesFilePath: string
2455: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2456: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2457: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2458: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2459: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2460: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2461: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2462: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2463: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2464: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2465: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2466: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2467: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2468: Function SetCurrentDir( Dir : string) : Boolean
2469: function SetCurrentDir(const Dir: string): Boolean
2470: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2471: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2472: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2473: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2474: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2475: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2476: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2477: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2478: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2479: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2480: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2481: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2482: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2483: Function SetLocalTime( Value : TDateTime) : boolean
2484: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2485: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2486: Function SetPrivilege(privilegeName: string; enable: boolean): boolean
2487: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2488: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2489: Function SetSize( libNewSize : Longint) : HResult
2490: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2491: Function Sgn( const X : Extended) : Integer
2492: function SHA1(const fileName: string): string;
2493: function SHA256(astr: string; amode: char): string
2494: function SHA512(astr: string; amode: char): string
2495: Function ShareMemoryManager : Boolean
2496: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2497: function Shelleexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2498: Function Shelleexecute3(afilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2499: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2500: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2501: function ShortDateFormat: string;
2502: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
    RTL:Bool;EllipsisWidth:Int):WideString
2503: function ShortTimeFormat: string;
2504: function SHOWMODAL:INTEGER
2505: Function ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:
    TColor; BW: Integer;Title:String;BeforeShowModal:TNotifyEvent): TModalResult';
2506: Function
    ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2507: function ShowWindow(C1: HWND; C2: integer): boolean;
2508: procedure ShowMemory //in Dialog
2509: function ShowMemory2: string;
2510: Function ShutDownOS : Boolean
2511: Function Signe( const X, Y : Extended) : Extended
2512: Function Sign( const X : Extended) : Integer
2513: Function Sin(e : Extended) : Extended;
2514: Function sinc( const x : Double) : Double
2515: Function SinJ( X : Float) : Float
2516: Function Size( const AFileName : String) : Integer
2517: function SizeOf: Longint;
2518: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2519: function SlashSep(const Path, S: String): String
2520: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2521: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2522: Function SmallPoint(X, Y: Integer): TSmallPoint
2523: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2524: Function SoundexCompare( const ATText : string; AOther : string; ALength : TSoundexLength) : Integer
2525: Function SoundexInt( const ATText : string; ALength : TSoundexIntLength) : Integer
2526: Function SoundexProc( const ATText, AOther : string) : Boolean
2527: Function SoundexSimilar( const ATText, AOther : string; ALength : TSoundexLength) : Boolean
2528: Function SoundexWord( const ATText : string) : Word
2529: Function SourcePos : Longint
2530: function SourcePos:LongInt

```

```

2531: Function Split0( Str : string; const substr : string) : TStringList
2532: Procedure SplitNameValuePair( const Line : string; var Name, Value : string)
2533: Function SQLRequiresParams( const SQL : WideString) : Boolean
2534: Function Sqr(e : Extended) : Extended;
2535: Function Sqrt(e : Extended) : Extended;
2536: Function StartIP : String
2537: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2538: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2539: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2540: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2541: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2542: Function StartOfAYear( const AYear : Word) : TDateTime
2543: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2544: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2545: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2546: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2547: Function StartsStr( const ASubText, AText : string) : Boolean
2548: Function StartsText( const ASubText, AText : string) : Boolean
2549: Function StartsWith( const ANSIString, APattern : String) : Boolean
2550: Function StartsWith( const str : string; const sub : string) : Boolean
2551: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2552: Function StatusString( StatusCode : Integer) : string
2553: Function StdDev( const Data : array of Double) : Extended
2554: Function Stop : Float
2555: Function StopCount( var Counter : TJclCounter) : Float
2556: Function StoreColumns : Boolean
2557: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2558: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2559: Function StrAlloc( Size : Cardinal) : PChar
2560: function StrAlloc(Size: Cardinal): PChar
2561: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2562: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2563: Function StrBufSize( Str : PChar) : Cardinal
2564: function StrBufSize(const Str: PChar): Cardinal
2565: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2566: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType
2567: Function StrCat( Dest : PChar; Source : PChar) : PChar
2568: function StrCat(Dest: PChar; const Source: PChar): PChar
2569: Function StrCharLength( Str : PChar) : Integer
2570: Function StrComp( Str1, Str2 : PChar) : Integer
2571: function StrComp(const Str1: PChar; const Str2: PChar): Integer
2572: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2573: function StrCopy(Dest: PChar; const Source: PChar): PChar
2574: Function Stream_to_AnsiString( Source : TStream) : ansistring
2575: Function Stream_to_Base64( Source : TStream) : ansistring
2576: Function Stream_to_decimalbytes( Source : TStream) : string
2577: Function Stream2WideString( oStream : TStream) : WideString
2578: Function StreamtoAnsiString( Source : TStream) : ansistring
2579: Function StreamToByte( Source : TStream) : string
2580: Function Stream.ToDecimalbytes( Source : TStream) : string
2581: Function StreamtoOrd( Source : TStream) : string
2582: Function StreamToString( Source : TStream) : string
2583: Function StreamToString2( Source : TStream) : string
2584: Function StreamToString3( Source : TStream) : string
2585: Function StreamToString4( Source : TStream) : string
2586: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2587: Function StrEmpty( const sString : string) : boolean
2588: Function StrEnd( Str : PChar) : PChar
2589: function StrEnd(const Str: PChar): PChar
2590: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2591: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar
2592: Function StrGet(var S : String; I : Integer) : Char;
2593: Function StrGet2(S : String; I : Integer) : Char;
2594: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2595: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2596: Function StrHtmlDecode( const AStr : String) : String
2597: Function StrHtmlEncode( const AStr : String) : String
2598: Function StrToBytes(const Value: String): TBytes;
2599: Function StrICmp( Str1, Str2 : PChar) : Integer
2600: Function StringOfChar(c : char; I : longInt) : String;
2601: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2602: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2603: Function StringRefCount(const s: String): integer;
2604: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2605: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2606: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2607: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2608: Function StringToBoolean( const Ps : string) : Boolean
2609: function StringToColor(const S: string): TColor
2610: function StringToCursor(const S: string): TCursor;
2611: function StringToGUID(const S: string): TGUID)
2612: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2613: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2614: Function StringWidth( S : string) : Integer
2615: Function StrInternetToDate( Value : string) : TDateTime
2616: Function StrIsDateTime( const Ps : string) : Boolean
2617: Function StrIsFloatMoney( const Ps : string) : Boolean
2618: Function StrIsInteger( const S : string) : Boolean
2619: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar

```

```

2620: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal ) : Integer
2621: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal ) : PChar
2622: Function StrLen( Str : PChar) : Cardinal
2623: function StrLen(const Str: PChar): Cardinal
2624: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2625: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2626: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal ) : Integer
2627: Function StrLower( Str : PChar) : PChar
2628: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal ) : PChar
2629: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar
2630: Function StrNew( Str : PChar) : PChar
2631: function StrNew(const Str: PChar): PChar
2632: Function StrNextChar( Str : PChar) : PChar
2633: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2634: Function StrParse( var sString : string; const sDelimiters : string) : string;
2635: Function StrParseSel( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2636: Function StrPas( Str : PChar) : string
2637: function StrPas(const Str: PChar): string
2638: Function StrPCopy( Dest : PChar; Source : string) : PChar
2639: function StrPCopy(Dest: PChar; const Source: string): PChar
2640: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal ) : PChar
2641: Function StrPos( Str1, Str2 : PChar) : PChar
2642: Function StrScan(const Str: PChar; Chr: Char): PChar)
2643: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2644: Function StrToBcd( const AValue : string) : TBcd
2645: Function StrToBool( S : string) : Boolean
2646: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2647: Function StrToCard( const AStr : String) : Cardinal
2648: Function StrToConv( AText : string; out AType : TConvType) : Double
2649: Function StrToCurr( S : string) : Currency;
2650: function StrToCurr(const S: string): Currency)
2651: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2652: Function StrToDate( S : string) : TDateTime;
2653: function StrToDate(const s: string): TDateTime;
2654: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2655: Function StrToDateDefTime( S : string) : TDateTime;
2656: function StrToDateDefTime(const S: string): TDateTime)
2657: Function StrToDateDefTimeDef( S : string; Default : TDateTime) : TDateTime;
2658: Function StrToDay( const ADay : string) : Byte
2659: Function StrtoFloat( S : string) : Extended;
2660: function StrtoFloat(s: String): Extended;
2661: Function StrtoFloatDef( S : string; Default : Extended) : Extended;
2662: function StrtoFloatDef(const S: string; const Default: Extended): Extended
2663: Function StrtoFloat( S : string) : Extended;
2664: Function StrtoFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2665: Function StrtoFloatDef( S : string; Default : Extended) : Extended;
2666: Function StrtoFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2667: Function StrToCurr( S : string) : Currency;
2668: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2669: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2670: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2671: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2672: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2673: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2674: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2675: Function StrToDateTime( S : string) : TDateTime;
2676: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2677: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2678: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2679: Function StrToInt( S : string) : Integer
2680: function StrToInt(s: String): Longint;
2681: Function StrToInt64( S : string) : Int64
2682: function StrToInt64(s: String): int64;
2683: Function StrToInt64Def( S : string; Default : Int64) : Int64
2684: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2685: Function StrToIntDef( S : string; Default : Integer) : Integer
2686: function StrToIntDef(const S: string; Default: Integer): Integer)
2687: function StrToIntDef(s: String; def: Longint): Longint;
2688: Function StrToMonth( const AMonth : string) : Byte
2689: Function StrToTime( S : string) : TDateTime;
2690: function StrToTime(const S: string): TDateTime)
2691: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2692: Function StrToWord( const Value : String) : Word
2693: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2694: Function StrToXmlDateTime( const DateStr: string; const Format : string) : string
2695: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2696: Function StrUpper( Str : PChar) : PChar
2697: Function StuffString( const AText : string; AStart, ALen : Cardinal; const ASubText : string) : string
2698: Function Sum( const Data : array of Double) : Extended
2699: Function SumFloatArray( const B : TDynFloatArray) : Float
2700: Function SumInt( const Data : array of Integer) : Integer
2701: Function SumOfSquares( const Data : array of Double) : Extended
2702: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2703: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2704: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2705: Function Supports( CursorOptions : TCursorOptions) : Boolean
2706: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2707: Function SwapWord(w : word): word)
2708: Function SwapInt(i : integer): integer)

```

```

2709: Function SwapLong(L : longint): longint
2710: Function Swap(i : integer): integer
2711: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2712: Function SyncTime : Boolean
2713: Function SysErrorMessage( ErrorCode : Integer) : string
2714: function SysErrorMessage(ErrorCode: Integer): string
2715: Function SystemTimeToDate( SystemTime : TSystemTime) : TDateTime
2716: function SystemTimeToDate( const SystemTime: TSystemTime): TDateTime;
2717: Function SysStringLen(const S: WideString): Integer; stdcall;
2718: Function TabRect( Index : Integer) : TRect
2719: Function Tan( const X : Extended) : Extended
2720: Function TaskMessageDlg(const Title,
Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2721: Function TaskMessageDlg( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2722: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
HelpCtx : Longint; X, Y : Integer) : Integer;
2723: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2724: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2725: Function TaskMessageDlgPosHelp1( const Title, Msg:string;DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2726: Function TenToY( const Y : Float) : Float
2727: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2728: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2729: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2730: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2731: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2732: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2733: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2734: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2735: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2736: Function TestBits( const Value, Mask : Byte) : Boolean;
2737: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2738: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2739: Function TestBits3( const Value, Mask : Word) : Boolean;
2740: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2741: Function TestBits5( const Value, Mask : Integer) : Boolean;
2742: Function TestBits6( const Value, Mask : Int64) : Boolean;
2743: Function TestFDIVInstruction : Boolean
2744: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2745: Function TextExtent( const Text: string) : TSize
2746: function TextHeight(Text: string): Integer
2747: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2748: Function TextStartsWith( const S, SubS : string) : Boolean
2749: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatingValue): Boolean)
2750: Function ConvInteger(i : integer):string;
2751: Function IntegerToText(i : integer):string;
2752: Function TEXTTOSHORTCUT( TEXT : String) : TSHORCUT
2753: function TextWidth(Text: string): Integer;
2754: Function ThreadCount : integer
2755: function ThousandSeparator: char;
2756: Function Ticks : Cardinal
2757: Function Time : TDateTime
2758: function Time: TDateTime;
2759: function TimeGetTime: int64;
2760: Function TimeOf( const AValue : TDateTime) : TDateTime
2761: function TimeSeparator: char;
2762: functionTimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2763: FunctionTimeStampToMSECS( TimeStamp : TTimeStamp ) : Comp
2764: functionTimeStampToMSECS(const TimeStamp: TTimeStamp): Comp
2765: Function TimeToStr( DateTime : TDateTime) : string;
2766: function TimeToStr(const DateTime: TDateTime): string;
2767: Function TimeZoneBias : TDateTime
2768: Function ToCommon( const AValue : Double) : Double
2769: function ToCommon(const AValue: Double): Double;
2770: Function Today : TDateTime
2771: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2772: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2773: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2774: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2775: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2776: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2777: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2778: function TokenComponentIdent:string
2779: Function TokenFloat : Extended
2780: function TokenFloat:Extended
2781: Function TokenInt : Longint
2782: function TokenInt:LongInt
2783: Function TokenString : string
2784: function TokenString:String
2785: Function TokenSymbolIs( const S : string) : Boolean
2786: function TokenSymbolIs(S:String):Boolean
2787: Function Tomorrow : TDateTime
2788: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2789: Function ToString : string
2790: Function TotalVariance( const Data : array of Double) : Extended
2791: Function Trace2( AURL : string ) : string;

```

```

2792: Function TrackMenu( Button : TToolButton ) : Boolean
2793: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN ) : INTEGER
2794: Function TranslateURI( const URI : string ) : string
2795: Function TranslationMatchesLanguages( Exact : Boolean ) : Boolean
2796: Function TransparentStretchBlt( DstDC : HDC; DstX, DstY, DstW, DstH : Integer; SrcDC : HDC; SrcX, SrcY, SrcW, SrcH : Integer; MaskDC : HDC; MaskX, MaskY : Integer ) : Boolean
2797: Function Trim( S : string ) : string
2798: Function Blank( S : string ) : string; //alias to Trim
2799: Function Trim( S : WideString ) : WideString;
2800: Function Trim(s : AnyString) : AnyString;
2801: Function TrimAllOf( ATrim, AText : String ) : String
2802: Function TrimLeft( S : string ) : string;
2803: Function TrimLeft(const S: string): string)
2804: function TrimLeft(const S: string): string)
2805: Function TrimRight( S : string ) : string;
2806: Function TrimRight( S : WideString ) : WideString;
2807: function TrimRight(const S: string): string)
2808: function TrueBoolStrs: array of string
2809: Function Trunc(e : Extended) : Longint;
2810: Function Trunc64(e: extended): Int64;
2811: Function TruncPower( const Base, Exponent : Float ) : Float
2812: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily ) : Boolean
2813: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily ) : Boolean;
2814: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean
2815: Function TryEncodeDateDay( const AYear, ADayOfYear: Word; out AValue : TDateTime ) : Boolean
2816: Function TryEncodeDateMonthWeek( const AY, AMonth, AWeekOfMonth, ADayOfWeek:Word;var AValue:TDateTime): Boolean
2817: Function TryEncodeDateTime( const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out AValue:TDateTime):Boolean
2818: Function TryEncodeDateWeek( const AY, AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2819: Function TryEncodeDayOfWeekInMonth( const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out AVal:TDateTime):Bool
2820: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2821: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime ) : Boolean
2822: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime ) : Boolean
2823: Function TryLock : Boolean
2824: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime ) : Boolean
2825: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond : Word; out AResult : TDateTime ) : Boolean
2826: Function TryStrToBcd( const AValue : string; var Bcd : TBcd ) : Boolean
2827: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType ) : Boolean
2828: Function TryStrToDate( S : string; Value : TDateTime ) : Boolean;
2829: Function TryStrToDate( S : string; Value : TDateTime ) : Boolean;
2830: Function TryStrToTime( S : string; Value : TDateTime ) : Boolean;
2831: Function TryStrToInt( const S: AnsiString; var I: Integer): Boolean;
2832: Function TryStrToInt64( const S: AnsiString; var I: Int64): Boolean;
2833: function TryStrToBool( const S: string; out Value: Boolean): Boolean;
2834: Function TwoByteToWord( AByte1, AByte2 : Byte ) : Word
2835: Function TwoCharToWord( AChar1, AChar2 : Char ) : Word
2836: Function TwoToY( const Y : Float ) : Float
2837: Function UCS4StringToWideString( const S : UCS4String ) : WideString
2838: Function UIDL( const ADest : TStrings; const AMsgNum : Integer ) : Boolean
2839: function Unassigned: Variant;
2840: Function UndoLastChange( FollowChange : Boolean ) : Boolean
2841: function UniCodeToStr( Value: string): string;
2842: Function UnionRect( out Rect : TRect; const R1, R2 : TRect ) : Boolean
2843: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2844: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal ) : TDateTime
2845: Function UnixPathToDosPath( const Path : string ) : string
2846: Function UnixToDateTime( const AValue : Int64 ) : TDateTime
2847: function UnixToDateTime(U: Int64): TDateTime;
2848: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint ) : HResult
2849: Function UnlockResource( ResData: HGLOBAL ) : LongBool
2850: Function UnlockVolume( var Handle : THandle ) : Boolean
2851: Function UnMaskString( Mask, Value : String ) : String
2852: function UpCase(ch : Char ) : Char;
2853: Function UpCaseFirst( const AStr : string ) : string
2854: Function UpCaseFirstWord( const AStr : string ) : string
2855: Function UpdateAction( Action : TBasicAction ) : Boolean
2856: Function UpdateKind : TUUpdateKind
2857: Function UPDATESTATUS : TUUPDATESTATUS
2858: Function UpperCase( S : string ) : string
2859: Function Uppercase(s : AnyString) : AnyString;
2860: Function URLDecode( ASrc : string ) : string
2861: Function URLEncode( const ASrc : string ) : string
2862: Function UseRightToLeftAlignment : Boolean
2863: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment ) : Boolean
2864: Function UseRightToLeftReading : Boolean
2865: Function UTF8CharLength( Lead : Char ) : Integer
2866: Function UTF8CharSize( Lead : Char ) : Integer
2867: Function UTF8Decode( const S : UTF8String ) : WideString
2868: Function UTF8Encode( const WS : WideString ) : UTF8String
2869: Function UTF8LowerCase( const S : UTF8String ) : UTF8String
2870: Function Utf8ToAnsi( const S : UTF8String ) : string
2871: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer ) : string
2872: Function UTF8UpperCase( const S : UTF8String ) : UTF8String
2873: Function ValidFieldIndex( FieldIndex : Integer ) : Boolean
2874: Function ValidParentForm(control: TControl): TForm
2875: Function Value : Variant
2876: Function ValueExists( const Section, Ident : string ) : Boolean

```

```

2877: Function ValueOf( const Key : string ) : Integer
2878: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2879: Function VALUEOFKEY( const AKEY : VARIANT ) : VARIANT
2880: Function VarArrayFromStrings( Strings : TStrings ) : Variant
2881: Function VarArrayFromWideStrings( Strings : TWideStrings ) : Variant
2882: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2883: Function VarFMTBcd : TVarType
2884: Function VarFMTBcdCreate1 : Variant;
2885: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word ) : Variant;
2886: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word ) : Variant;
2887: Function VarFMTBcdCreate4( const ABcd : TBcd ) : Variant;
2888: Function Variance( const Data : array of Double ) : Extended
2889: Function VariantAdd2( const V1 : Variant; const V2 : Variant ) : Variant
2890: Function VariantAnd2( const V1 : Variant; const V2 : Variant ) : Variant
2891: Function VariantDiv2( const V1 : Variant; const V2 : Variant ) : Variant
2892: Function VariantgetElement( const V : Variant; i1 : integer ) : Variant;
2893: Function VariantgetElement1( const V : Variant; i1, i2 : integer ) : Variant;
2894: Function VariantgetElement2( const V : Variant; i1, i2, i3 : integer ) : Variant;
2895: Function VariantgetElement3( const V : Variant; i1, i2, i3, i4 : integer ) : Variant;
2896: Function VariantgetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer ) : Variant;
2897: Function VariantMod2( const V1 : Variant; const V2 : Variant ) : Variant
2898: Function VariantMul2( const V1 : Variant; const V2 : Variant ) : Variant
2899: Function VariantNeg( const V1 : Variant ) : Variant
2900: Function VariantNot( const V1 : Variant ) : Variant
2901: Function VariantOr2( const V1 : Variant; const V2 : Variant ) : Variant
2902: Function VariantShl2( const V1 : Variant; const V2 : Variant ) : Variant
2903: Function VariantShr2( const V1 : Variant; const V2 : Variant ) : Variant
2904: Function VariantSub2( const V1 : Variant; const V2 : Variant ) : Variant
2905: Function VariantXor2( const V1 : Variant; const V2 : Variant ) : Variant
2906: function VarIsEmpty(const V: Variant): Boolean;
2907: Function VarIsFMTBcd( const AValue : Variant ) : Boolean;
2908: function VarIsNull(const V: Variant): Boolean;
2909: Function VarToBcd( const AValue : Variant ) : TBcd
2910: function VarType(const V: Variant): TVarType;
2911: Function VarType( const V : Variant ) : TVarType
2912: Function VarAsType( const V : Variant; AVarType : TVarType ) : Variant
2913: Function VarIsType( const V : Variant; AVarType : TVarType ) : Boolean;
2914: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType ) : Boolean;
2915: Function VarIsByRef( const V : Variant ) : Boolean
2916: Function VarIsEmpty( const V : Variant ) : Boolean
2917: Procedure VarCheckEmpty( const V : Variant )
2918: Function VarIsNull( const V : Variant ) : Boolean
2919: Function VarIsClear( const V : Variant ) : Boolean
2920: Function VarIsCustom( const V : Variant ) : Boolean
2921: Function VarIsOrdinal( const V : Variant ) : Boolean
2922: Function VarIsFloat( const V : Variant ) : Boolean
2923: Function VarIsNumeric( const V : Variant ) : Boolean
2924: Function VarIsStr( const V : Variant ) : Boolean
2925: Function VarToStr( const V : Variant ) : string
2926: Function VarToStrDef( const V : Variant; const ADefault : string ) : string
2927: Function VarToWideStr( const V : Variant ) : WideString
2928: Function VarToWideStrDef( const V : Variant; const ADefault : WideString ) : WideString
2929: Function VarToDate( const V : Variant ) : TDate
2930: Function VarFromDate( const Date : TDate ) : Variant
2931: Function VarInRange( const AValue, AMin, AMax : Variant ) : Boolean
2932: Function VarEnsureRange( const AValue, AMin, AMax : Variant ) : Variant
2933: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2934: Function VarSameValue( const A, B : Variant ) : Boolean
2935: Function VarCompareValue( const A, B : Variant ) : TVariantRelationship
2936: Function VarIsEmptyParam( const V : Variant ) : Boolean
2937: Function VarIsError( const V : Variant; out AResult : HRESULT ) : Boolean;
2938: Function VarIsError1( const V : Variant ) : Boolean;
2939: Function VarAsError( AResult : HRESULT ) : Variant
2940: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant )
2941: Function VarIsArray( const A : Variant ) : Boolean;
2942: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean ) : Boolean;
2943: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType ) : Variant
2944: Function VarArrayOf( const Values : array of Variant ) : Variant
2945: Function VarArrayRef( const A : Variant ) : Variant
2946: Function VarTypeIsValidArrayType( const AVarType : TVarType ) : Boolean
2947: Function VarTypeIsValidElementType( const AVarType : TVarType ) : Boolean
2948: Function VarArrayDimCount( const A : Variant ) : Integer
2949: Function VarArrayLowBound( const A : Variant; Dim : Integer ) : Integer
2950: Function VarArrayHighBound( const A : Variant; Dim : Integer ) : Integer
2951: Function VarArrayLock( const A : Variant ) : __Pointer
2952: Procedure VarArrayUnlock( const A : Variant )
2953: Function VarArrayGet( const A : Variant; const Indices : array of Integer ) : Variant
2954: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer )
2955: Procedure DynArrayToVariant( var V : Variant; const DynArray : __Pointer; TypeInfo : __Pointer )
2956: Procedure DynArrayFromVariant( var DynArray : __Pointer; const V : Variant; TypeInfo : __Pointer )
2957: Function Unassigned : Variant
2958: Function Null : Variant
2959: Function VectorAdd( const V1, V2 : TFlopPoint ) : TFlopPoint
2960: function VectorAdd(const V1,V2: TFlopPoint): TFlopPoint;
2961: Function VectorDot( const V1, V2 : TFlopPoint ) : Double
2962: function VectorDot(const V1,V2: TFlopPoint): Double;
2963: Function VectorLengthSqr( const V : TFlopPoint ) : Double
2964: function VectorLengthSqr(const V: TFlopPoint): Double;
2965: Function VectorMult( const V : TFlopPoint; const s : Double ) : TFlopPoint

```

```

2966: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2967: Function VectorSubtract( const V1, V2 : TFloatPoint ) : TFloatPoint
2968: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2969: Function Verify( AUserName : String ) : String
2970: Function Versine( X : Float ) : Float
2971: function VersionCheck: boolean;
2972: function VersionCheckAct: string;
2973: Function VersionLanguageId( const LangIdRec : TLangIdRec ) : string
2974: Function VersionLanguageName( const LangId : Word ) : string
2975: Function VersionResourceAvailable( const FileName : string ) : Boolean
2976: Function Visible : Boolean
2977: function VolumeID(DriveChar: Char): string
2978: Function WaitFor( const AString : string ) : string
2979: Function WaitFor( const TimeOut : Cardinal ) : TJclWaitResult
2980: Function WaitForI : TWaitResult;
2981: Function WaitForData( Timeout : Longint ) : Boolean
2982: Function WebColorNameToColor( WebColorName : string ) : TColor
2983: Function WebColorStrToColor( WebColor : string ) : TColor
2984: Function WebColorToRGB( WebColor : Integer ) : Integer
2985: Function wGet(aURL, afile: string): boolean;
2986: Function wGet2(aURL, afile: string): boolean; //without file open
2987: Function wGetX(aURL, afile: string): boolean;
2988: Function wGetX2(aURL, afile: string): boolean; //without file open
2989: Function WebGet(aURL, afile: string): boolean;
2990: Function WebExists: boolean; //alias to isinternet
2991: Function WeekOf( const AValue : TDateTime ) : Word
2992: Function WeekOfTheMonth( const AValue : TDateTime ) : Word;
2993: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word ) : Word;
2994: Function WeekOfTheYear( const AValue : TDateTime ) : Word;
2995: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word ) : Word;
2996: Function WeeksBetween( const ANow, AThen : TDateTime ) : Integer
2997: Function WeeksInAYear( const AYear : Word ) : Word
2998: Function WeeksInYear( const AValue : TDateTime ) : Word
2999: Function WeekSpan( const ANow, AThen : TDateTime ) : Double
3000: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle ) : WideString
3001: Function WideCat( const x, y : WideString ) : WideString
3002: Function WideCompareStr( S1, S2 : WideString ) : Integer
3003: function WideCompareStr(const S1: WideString; const S2: WideString): Integer
3004: Function WideCompareText( S1, S2 : WideString ) : Integer
3005: function WideCompareText(const S1: WideString; const S2: WideString): Integer
3006: Function WideCopy( const src : WideString; index, count : Integer ) : WideString
3007: Function WideDequotedStr( const S : WideString; AQuote : WideChar ) : WideString
3008: Function WideEqual( const x, y : WideString ) : Boolean
3009: function WideFormat(const Format: WideString; const Args: array of const): WideString)
3010: Function WideGreater( const x, y : WideString ) : Boolean
3011: Function WideLength( const src : WideString ) : Integer
3012: Function WideLess( const x, y : WideString ) : Boolean
3013: Function WideLowerCase( S : WideString ) : WideString
3014: function WidLowerCase(const S: WideString): WideString
3015: Function WidePos( const src, sub : WideString ) : Integer
3016: Function WideQuotedStr( const S : WideString; Quote : WideChar ) : WideString
3017: Function WideReplaceStr( const AText, AFromText, AToText : WideString ) : WideString
3018: Function WideReplaceText( const AText, AFromText, AToText : WideString ) : WideString
3019: Function WideSameStr( S1, S2 : WideString ) : Boolean
3020: function WideSameStr(const S1: WideString; const S2: WideString): Boolean
3021: Function WideSameText( S1, S2 : WideString ) : Boolean
3022: function WideSameText(const S1: WideString; const S2: WideString): Boolean
3023: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
3024: Function WideStringToUCS4String( const S : WideString ) : UCS4String
3025: Function WideUpperCase( S : WideString ) : WideString
3026: Function Win32BackupFile( const FileName : string; Move : Boolean ) : Boolean
3027: function Win32Check(RetVal: boolean): boolean
3028: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
3029: Function Win32RestoreFile( const FileName : string ) : Boolean
3030: Function Win32Type : TIdWin32Type
3031: Function WinColor( const Color32 : TColor32 ) : TColor
3032: function winexec(FileNamed: pchar; showCmd: integer): integer;
3033: Function WinExec32( const Cmd : string; const CmdShow : Integer ) : Boolean
3034: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer ) : Cardinal
3035: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer ) : Boolean
3036: Function WithinPastHours( const ANow, ATThen : TDateTime; const AHours : Int64 ) : Boolean
3037: Function WithinPastMilliSeconds( const ANow, ATThen : TDateTime; const AMilliSeconds : Int64 ) : Boolean
3038: Function WithinPastMinutes( const ANow, ATThen : TDateTime; const AMinutes : Int64 ) : Boolean
3039: Function WithinPastMonths( const ANow, ATThen : TDateTime; const AMonths : Integer ) : Boolean
3040: Function WithinPastSeconds( const ANow, ATThen : TDateTime; const ASeconds : Int64 ) : Boolean
3041: Function WithinPastWeeks( const ANow, ATThen : TDateTime; const AWeeks : Integer ) : Boolean
3042: Function WithinPastYears( const ANow, ATThen : TDateTime; const AYears : Integer ) : Boolean
3043: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar ) : DWORD
3044: Function WordToStr( const Value : Word ) : String
3045: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint ) : Boolean
3046: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string ) : Boolean
3047: Procedure GetWordGridFormatValues( Proc : TGetStrProc )
3048: Function WorkArea : Integer
3049: Function WrapText( Line : string; MaxCol : Integer ) : string;
3050: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer ) : string;
3051: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint ) : HResult
3052: function Write(Buffer:String;Count:LongInt):LongInt
3053: Function WriteClient( var Buffer, Count : Integer ) : Integer
3054: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean ) : Cardinal

```

```

3055: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
3056: Function WriteString( const AString : string) : Boolean
3057: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
3058: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
3059: Function wsprintf( Output : PChar; Format : PChar) : Integer
3060: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
3061: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
3062: Function XorDecode( const Key, Source : string) : string
3063: Function XorEncode( const Key, Source : string) : string
3064: Function XorString( const Key, Src : ShortString) : ShortString
3065: Function Yield : Bool
3066: Function YearOf( const AValue : TDateTime) : Word
3067: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3068: Function YearSpan( const ANow, AThen : TDateTime) : Double
3069: Function Yesterday : TDateTime
3070: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3071: Function( const Name : string; Proc : TUserFunction)
3072: Function using Special_Scholz from 3.8.5.0
3073: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3074: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3075: Function FloatToTime2Dec(value:Extended):Extended;
3076: Function MinToStd(value:Extended):Extended;
3077: Function MinToStdAsString(value:Extended):String;
3078: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3079: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3080: Function Round2Dec (zahl:Extended):Extended;
3081: Function GetAngle(x,y:Extended):Double;
3082: Function AddAngle(a1,a2:Double):Double;
3083:
3084: ****
3085: unit UPST_StText;
3086: ****
3087: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3088: Function TextFileSize( var F : TextFile) : LongInt
3089: Function TextPos( var F : TextFile) : LongInt
3090: Function TextFlush( var F : TextFile) : Boolean
3091:
3092: ****
3093: from JvVCLUtils;
3094: ****
3095: { Windows resources (bitmaps and icons) VCL-oriented routines }
3096: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3097: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
  DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3098: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
  Bitmap: TBitmap; TransparentColor: TColor);
3099: function MakeBitmap(ResID: PChar): TBitmap;
3100: function MakeBitmapID(ResID: Word): TBitmap;
3101: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3102: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3103: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3104: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
  HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3105: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3106: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3107: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3108:
3109: {$IFDEF WIN32}
3110: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
  X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3112: {$ENDIF}
3113: function MakeIcon(ResID: PChar): TIcon;
3114: function MakeIconID(ResID: Word): TIcon;
3115: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3116: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3117: {$IFDEF WIN32}
3118: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3119: {$ENDIF}
3120: { Service routines }
3121: procedure NotImplemented;
3122: procedure ResourceNotFound(ResID: PChar);
3123: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3124: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3125: function PaletteColor(Color: TColor): Longint;
3126: function WidthOf(R: TRect): Integer;
3127: function HeightOf(R: TRect): Integer;
3128: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3129: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3130: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3131: procedure Delay(MSecs: Longint);
3132: procedure DeleteLine(StrList: TStringList; SearchPattern: String);
3133: procedure CenterControl(Control: TControl);
3134: Function PaletteEntries( Palette : HPALETTE ) : Integer
3135: Function WindowClassName( Wnd : HWND ) : string
3136: Function ScreenWorkArea : TRect
3137: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3138: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3139: Procedure ActivateWindow( Wnd : HWND )
3140: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3141: Procedure CenterWindow( Wnd : HWND )

```

```

3142: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3143: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3144: Function DialogsToPixelsX( Dlgs : Word) : Word
3145: Function DialogsToPixelsY( Dlgs : Word) : Word
3146: Function PixelsToDialogsX( Pixs : Word) : Word
3147: Function PixelsToDialogsY( Pixs : Word) : Word
3148: {$IFDEF WIN32}
3149: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3150: function MakeVariant(const Values: array of Variant): Variant;
3151: {$ENDIF}
3152: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3153: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3154: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3155: {$IFDEF CBUILDERS}
3156: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3157: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3158: {$ELSE}
3159: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3160: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3161: {$ENDIF CBUILDERS}
3162: function IsForegroundTask: Boolean;
3163: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3164: function GetAveCharSize(Canvas: TCanvas): TPoint;
3165: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3166: procedure FreeUnusedOLE;
3167: procedure Beep;
3168: function GetWindowsVersionJ: string;
3169: function LoadDLL(const LibName: string): THandle;
3170: function RegisterServer(const ModuleName: string): Boolean;
3171: {$IFDEF WIN32}
3172: function IsLibrary: Boolean;
3173: {$ENDIF}
3174: { Gradient filling routine }
3175: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3176: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction: TFillDirection; Colors: Byte);
3177: { String routines }
3178: function GetEnvVar(const VarName: string): string;
3179: function AnsiUpperFirstChar(const S: string): string;
3180: function StringToPChar(var S: string): PChar;
3181: function StrPAalloc(const S: string): PChar;
3182: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
3183: function DropT(const S: string): string;
3184: { Memory routines }
3185: function AllocMemo(Size: Longint): Pointer;
3186: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3187: procedure FreeMemo(var fpBlock: Pointer);
3188: function GetMemoSize(fpBlock: Pointer): Longint;
3189: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3190: {$IFDEF COMPILERS_UP}
3191: procedure FreeAndNil(var Obj);
3192: {$ENDIF}
3193: // from PNGLoader
3194: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3195: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3196: Procedure TransformLOC02RGB( Image : TLinearBitmap)
3197: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3198: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style : TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect //Buttons
3199: Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3200: Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3201: AddConstantN('CTL3D_ALL','LongWord').SetUInt($FFFF);
3202: //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3203: //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3204: //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3205: Procedure SetIMEMode( hWnd : HWND; Mode : TImeMode)
3206: Procedure SetIMEName( Name : TImeName)
3207: Function Win32NLEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3208: Function Imm32GetContext( hWnd : HWND) : HIMC
3209: Function Imm32ReleaseContext( hWnd : HWND; himc : HIMC) : Boolean
3210: Function Imm32GetConversionStatus( himc : HIMC; var Conversion, Sentence : longword) : Boolean
3211: Function Imm32SetConversionStatus( himc : HIMC; Conversion, Sentence : longword) : Boolean
3212: Function Imm32SetOpenStatus( himc : HIMC; fOpen : Boolean) : Boolean
3213: // Function Imm32SetCompositionWindow( himc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3214: //Function Imm32SetCompositionFont( himc : HIMC; lpLogfont : PLOGFONT) : Boolean
3215: Function Imm32GetCompositionString(himc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3216: Function Imm32IsIME( hkl : longword) : Boolean
3217: Function Imm32NotifyIME( himc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3218: Procedure DragDone( Drop : Boolean)
3219:
3220:
3221: //*****added from jvvcutils
3222: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3223: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3224: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3225: function IsPositiveResult(Value: TModalResult): Boolean;
3226: function IsNegativeResult(Value: TModalResult): Boolean;
3227: function IsAbortResult(const Value: TModalResult): Boolean;
3228: function StripAllFromResult(const Value: TModalResult): TModalResult;

```

```

3229: // returns either BrightColor or DarkColor depending on the luminance of AColor
3230: // This function gives the same result (AFAIK) as the function used in Windows to
3231: // calculate the desktop icon text color based on the desktop background color
3232: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor);
3233: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3234:
3235: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3236:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3237:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3238:   var LinkName: string; Scale: Integer = 100); overload;
3239: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3240:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3241:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3242:   var LinkName: string; Scale: Integer = 100); overload;
3243: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3244:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3245: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3246:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3247:   Scale: Integer = 100): string;
3248: function HTMLPlainText(const Text: string): string;
3249: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3250:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3251: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3252:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3253: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3254: function HTMLPrepareText(const Text: string): string;
3255:
3256: ***** uPSI_JvAppUtils;
3257: Function GetDefaultSection( Component : TComponent ) : string
3258: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3259: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3260: Function GetDefaultIniName : string
3261: //OnGetDefaultIniName,'TOnGetDefaultIniName';
3262: Function GetDefaultIniRegKey : string
3263: Function FindForm( FormClass : TFormClass ) : TForm
3264: Function FindShowForm( FormClass : TFormClass; const Caption : string ) : TForm
3265: Function ShowDialog( FormClass : TFormClass ) : Boolean
3266: //Function InstantiateForm( FormClass : TFormClass; var Reference ) : TForm
3267: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3268: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3269: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile )
3270: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string )
3271: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string )
3272: Function GetUniqueFileNameInDir( const Path, FileNameMask : string ) : string
3273: Function StrToIniStr( const Str : string ) : string
3274: Function IniStrToStr( const Str : string ) : string
3275: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string ) : string
3276: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string )
3277: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint ) : Longint
3278: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint )
3279: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean ) : Boolean
3280: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean )
3281: Procedure IniReadSections( IniFile : TObject; Strings : TStrings )
3282: Procedure IniEraseSection( IniFile : TObject; const Section : string )
3283: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string )
3284: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint )
3285: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint )
3286: Procedure AppTaskbarIcons( AppOnly : Boolean )
3287: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string )
3288: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string )
3289: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject )
3290: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject )
3291: ***** uPSI_JvDBUtils;
3292: Function CreateLocate( DataSet : TDataSet ) : TJvLocateObject
3293: Function IsDataSetEmpty( DataSet : TDataSet ) : Boolean
3294: Procedure RefreshQuery( Query : TDataSet )
3295: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3296: Function DataSetSectionName( DataSet : TDataSet ) : string
3297: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string )
3298: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3299: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
  Options: TLocateOptions) : Boolean
3300: Procedure SaveFields( DataSet : TDataSet; IniFile : TIIniFile )
3301: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIIniFile; RestoreVisible : Boolean )
3302: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean )
3303: Function ConfirmDelete : Boolean
3304: Procedure ConfirmDataSetCancel( DataSet : TDataSet )
3305: Procedure CheckRequiredField( Field : TField )
3306: Procedure CheckRequiredFields( const Fields : array of TField )
3307: Function DateToSQL( Value : TDateTime ) : string
3308: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string ) : string
3309: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string ) : string
3310: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
  HighEmpty:Double;Inclusive:Bool):string
3311: Function StrMaskSQL( const Value : string ) : string
3312: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3313: Function FormatAnsiSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3314: Procedure _DBError( const Msg : string )
3315: Const ('TrueExpr','String '0=0

```

```

3316: Const('sdfStandard16','String ''''''mm'/'dd'/'yyyy'''')
3317: Const('sdfStandard32','String ''''''dd/mm/yyyy'''')
3318: Const('sdfOracle','String ''TO_DATE('''dd/mm/yyyy''', ''DD/MM/YYYY'')')
3319: Const('sdfInterbase','String ''CAST('''mm''/'dd''/'yyyy''' AS DATE)'')
3320: Const('sdfMSSQL','String ''CONVERT(datetime, '''mm''/'dd''/'yyyy''' , 103)'')
3321: AddTypeS('Largeint','Longint')
3322: TIFEException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3323:   'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3324:   'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3325:   'erVersionError, ErDivideByZero, ErMathError, erCouldNotCallProc, erOutOfRecordRange, '+
3326:   'erOutOfMemory, erException, erNullPointerException, erNullVariantError, erInterfaceNotSupported, erDerError);
3327: (*-----*)
3328: procedure SIRегистер_JclIniFiles(CL: TPSFPascalCompiler);
3329: begin
3330:   Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3331:   Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3332:   Function JIniReadString( const FileName, Section, Line : string) : string
3333:   Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3334:   Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3335:   Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3336:   Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3337:   Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3338: end;
3339:
3340: (* == compile-time registration functions == *)
3341: (*-----*)
3342: procedure SIRегистер_JclDateTime(CL: TPSFPascalCompiler);
3343: begin
3344:   'UnixTimeStart','LongInt'( 25569 );
3345:   Function JEncodeDate( const Year : Integer; Month, Day : Word ) : TDateTime
3346:   Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word );
3347:   Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word );
3348:   Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer );
3349:   Function CenturyOfDate( const DateTime : TDateTime ) : Integer
3350:   Function CenturyBaseYear( const DateTime : TDateTime ) : Integer
3351:   Function DayOfDate( const DateTime : TDateTime ) : Integer
3352:   Function MonthOfDate( const DateTime : TDateTime ) : Integer
3353:   Function YearOfDate( const DateTime : TDateTime ) : Integer
3354:   Function JDdayOfTheYear( const DateTime : TDateTime; var Year : Integer ) : Integer;
3355:   Function DayOfTheYear1( const DateTime : TDateTime ) : Integer;
3356:   Function DayOfTheYearToDate( const Year, Day : Integer ) : TDateTime
3357:   Function HourOfTime( const DateTime : TDateTime ) : Integer
3358:   Function MinuteOfTime( const DateTime : TDateTime ) : Integer
3359:   Function SecondOfTime( const DateTime : TDateTime ) : Integer
3360:   Function GetISOYearNumberOfDay( const Year : Word ) : Word
3361:   Function IsISOLongYear( const Year : Word ) : Boolean;
3362:   Function IsISOLongYear1( const DateTime : TDateTime ) : Boolean;
3363:   Function ISODayOfWeek( const DateTime : TDateTime ) : Word
3364:   Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer ) : Integer;
3365:   Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer ) : Integer;
3366:   Function ISOWeekNumber2( DateTime : TDateTime ) : Integer;
3367:   Function ISOWeekToDate( const Year, Week, Day : Integer ) : TDateTime
3368:   Function JIsLeapYear( const Year : Integer ) : Boolean;
3369:   Function IsLeapYear1( const DateTime : TDateTime ) : Boolean;
3370:   Function JDdaysInMonth( const DateTime : TDateTime ) : Integer
3371:   Function Make4DigitYear( Year, Pivot : Integer ) : Integer
3372:   Function JMakeYear4Digit( Year, WindowsillYear : Integer ) : Integer
3373:   Function JEasterSunday( const Year : Integer ) : TDatetime // TDosDateTime', 'Integer
3374:   Function JFormatDateTime( Form : string; DateTime : TDateTime ) : string
3375:   Function FATDatesEqual( const FileTime1, FileTime2 : Int64 ) : Boolean;
3376:   Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime ) : Boolean;
3377:   Function HoursToMSecs( Hours : Integer ) : Integer
3378:   Function MinutesToMSecs( Minutes : Integer ) : Integer
3379:   Function SecondsToMSecs( Seconds : Integer ) : Integer
3380:   Function TimeOfDateToSeconds( DateTime : TDateTime ) : Integer
3381:   Function TimeOfDateToTimeToMSecs( DateTime : TDateTime ) : Integer
3382:   Function DateTimeToLocalDateTime( DateTime : TDateTime ) : TDateTime
3383:   Function LocalDateTimeToDate( DateTime : TDateTime ) : TDateTime
3384:   Function DateTimeToDosDateTime( const DateTime : TDateTime ) : TDosDateTime
3385:   Function JDDateToToFileTime( DateTime : TDateTime ) : TFileTime
3386:   Function JDDateToSystemTime( DateTime : TDateTime ) : TSystemTime;
3387:   Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime );
3388:   Function LocalDateTimeToFileTime( DateTime : TDateTime ) : FileTime
3389:   Function DosDateTimeToDate( const DosTime : TDosDateTime ) : TDateTime
3390:   Function JDosDateTimeToFileTime( DosTime : TDosDateTime ) : TFileTime;
3391:   Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime );
3392:   Function DosDateTimeToSystemTime( const DosTime : TDosDateTime ) : TSystemTime
3393:   Function DosDateTimeToStr( DateTime : Integer ) : string
3394:   Function JFileTimeToDate( const FileTime : TFileTime ) : TDateTime
3395:   Function FileTimeToLocalDateTime( const FileTime : TFileTime ) : TDateTime
3396:   Function JFileTimeToDosDateTime( const FileTime : TFileTime ) : TDosDateTime;
3397:   Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word );
3398:   Function JFileTimeToSystemTime( const FileTime : TFileTime ) : TSystemTime;
3399:   Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime );
3400:   Function FileTimeToStr( const FileTime : TFileTime ) : string
3401:   Function SystemTimeToDosDateTime( const SystemTime : TSystemTime ) : TDosDateTime
3402:   Function JSystemTimeToFileTime( const SystemTime : TSystemTime ) : TFileTime;
3403:   Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime );
3404:   Function SystemTimeToStr( const SystemTime : TSystemTime ) : string

```

```

3405: Function CreationDateTimeOfFile( const Sr : TSearchRec ) : TDateTime
3406: Function LastAccessDateTimeOfFile( const Sr : TSearchRec ) : TDateTime
3407: Function LastWriteDateTimeOfFile( const Sr : TSearchRec ) : TDateTime
3408: TJclUnixTime32', 'Longword
3409: Function JDateTimeToUnixTime( DateTime : TDateTime ) : TJclUnixTime32
3410: Function JUnixTimeToDate( const UnixTime : TJclUnixTime32 ) : TDateTime
3411: Function FileTimeToUnixTime( const AValue : TFileTime ) : TJclUnixTime32
3412: Function UnixTimeToFileTime( const AValue : TJclUnixTime32 ) : TFileTime
3413: Function JNullStamp : TTimeStamp
3414: Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Int64
3415: Function JEQualTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Boolean
3416: Function JIsNullTimeStamp( const Stamp : TTimeStamp ) : Boolean
3417: FunctionTimeStampDOW( const Stamp : TTimeStamp ) : Integer
3418: Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3419: Function FirstWeekDay1( const Year, Month : Integer ) : Integer;
3420: Function LastWeekDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3421: Function LastWeekDay1( const Year, Month : Integer ) : Integer;
3422: Function IndexedWeekDay( const Year, Month : Integer; Index : Integer ) : Integer
3423: Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3424: Function FirstWeekendDay1( const Year, Month : Integer ) : Integer;
3425: Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3426: Function LastWeekendDay1( const Year, Month : Integer ) : Integer;
3427: Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer ) : Integer
3428: Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer ) : Integer
3429: Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer ) : Integer
3430: Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer ) : Integer
3431: FindClass('TOBJECT'), EJclDateTimeError
3432: end;
3433:
3434: procedure SIRегистre_JclMiscel2(CL: TPSPPascalCompiler);
3435: begin
3436: Function SetDisplayResolution( const XRes, YRes : DWORD ) : Longint
3437: Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string ) : Boolean
3438: Function WinExec32( const Cmd : string; const CmdShow : Integer ) : Boolean
3439: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer ) : Cardinal
3440: Function WinExec32AndRedirectOutput( const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3441: TJclKillLevel', '( k1Normal, k1NoSignal, k1TimeOut )
3442: Function ExitWindows( ExitCode : Cardinal ) : Boolean
3443: Function LogOffOS( KillLevel : TJclKillLevel ) : Boolean
3444: Function PowerOffOS( KillLevel : TJclKillLevel ) : Boolean
3445: Function ShutDownOS( KillLevel : TJclKillLevel ) : Boolean
3446: Function RebootOS( KillLevel : TJclKillLevel ) : Boolean
3447: Function HibernateOS( Force, DisableWakeEvents : Boolean ) : Boolean
3448: Function SuspendOS( Force, DisableWakeEvents : Boolean ) : Boolean
3449: Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean ):Boolean;
3450: Function ShutDownDialog1( const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool ):Bool;
3451: Function AbortShutDown : Boolean;
3452: Function AbortShutDown1( const MachineName : string ) : Boolean;
3453: TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3454: TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3455: Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3456: FindClass('TOBJECT'), EJclCreateProcessError
3457: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string )
3458: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar);
3459: // with Add(EJclCreateProcessError) do
3460: end;
3461:
3462:
3463: procedure SIRегистre_JclAnsiStrings(CL: TPSPPascalCompiler);
3464: begin
3465: // 'AnsiSigns', 'Set').SetSet(['-', '+']);
3466: 'C1_UPPER', 'LongWord( $0001 );
3467: 'C1_LOWER', 'LongWord( $0002 );
3468: 'C1_DIGIT', 'LongWord').SetUInt( $0004 );
3469: 'C1_SPACE', 'LongWord').SetUInt( $0008 );
3470: 'C1_PUNCT', 'LongWord').SetUInt( $0010 );
3471: 'C1_CTRNL', 'LongWord').SetUInt( $0020 );
3472: 'C1_BLANK', 'LongWord').SetUInt( $0040 );
3473: 'C1_XDIGIT', 'LongWord').SetUInt( $0080 );
3474: 'C1_ALPHA', 'LongWord').SetUInt( $0100 );
3475: AnsiChar', 'Char
3476: Function StrIsAlpha( const S : AnsiString ) : Boolean
3477: Function StrIsAlphaNum( const S : AnsiString ) : Boolean
3478: Function StrIsAlphaNumUnderscore( const S : AnsiString ) : Boolean
3479: Function StrContainsChars( const S:AansiString;Chars:TSysCharSet; CheckAll : Boolean ) : Boolean
3480: Function StrConsistsOfNumberChars( const S : AnsiString ) : Boolean
3481: Function StrIsDigit( const S : AnsiString ) : Boolean
3482: Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet ) : Boolean
3483: Function StrSame( const S1, S2 : AnsiString ) : Boolean
3484: //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar ) : AnsiString
3485: Function StrCharPosLower( const S : AnsiString; CharPos : Integer ) : AnsiString
3486: Function StrCharPosUpper( const S : AnsiString; CharPos : Integer ) : AnsiString
3487: Function StrDoubleQuote( const S : AnsiString ) : AnsiString
3488: Function StrEnsureNoPrefix( const Prefix, Text : AnsiString ) : AnsiString
3489: Function StrEnsureNoSuffix( const Suffix, Text : AnsiString ) : AnsiString
3490: Function StrEnsurePrefix( const Prefix, Text : AnsiString ) : AnsiString
3491: Function StrEnsureSuffix( const Suffix, Text : AnsiString ) : AnsiString
3492: Function StrEscapedToString( const S : AnsiString ) : AnsiString

```

```

3493: Function JStrLower( const S : AnsiString ) : AnsiString
3494: Procedure StrLowerInPlace( var S : AnsiString )
3495: //Procedure StrLowerBuff( S : PAnsiChar )
3496: Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer );
3497: Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar ) : AnsiString
3498: Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar ) : AnsiString
3499: Function StrProper( const S : AnsiString ) : AnsiString
3500: //Procedure StrProperBuff( S : PAnsiChar )
3501: Function StrQuote( const S : AnsiString; C : AnsiChar ) : AnsiString
3502: Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3503: Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3504: Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3505: Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar ) : AnsiString
3506: Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3507: Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3508: Function StrRepeat( const S : AnsiString; Count : Integer ) : AnsiString
3509: Function StrRepeatLength( const S : AnsiString; const L : Integer ) : AnsiString
3510: Function StrReverse( const S : AnsiString ) : AnsiString
3511: Procedure StrReverseInPlace( var S : AnsiString )
3512: Function StrSingleQuote( const S : AnsiString ) : AnsiString
3513: Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet ) : AnsiString
3514: Function StrStringToEscaped( const S : AnsiString ) : AnsiString
3515: Function StrStripNonNumberChars( const S : AnsiString ) : AnsiString
3516: Function StrToHex( const Source : AnsiString ) : AnsiString
3517: Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar ) : AnsiString
3518: Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3519: Function StrTrimCharRight( const S : AnsiString; C : AnsiChar ) : AnsiString
3520: Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3521: Function StrTrimQuotes( const S : AnsiString ) : AnsiString
3522: Function JStrUpper( const S : AnsiString ) : AnsiString
3523: Procedure StrUpperInPlace( var S : AnsiString )
3524: //Procedure StrUpperBuff( S : PAnsiChar )
3525: Function StrOemToAnsi( const S : AnsiString ) : AnsiString
3526: Function StrAnsiToOem( const S : AnsiString ) : AnsiString
3527: Procedure StrAddRef( var S : AnsiString )
3528: Function StrAllocSize( const S : AnsiString ) : Longint
3529: Procedure StrDecRef( var S : AnsiString )
3530: //Function StrLen( S : PAnsiChar ) : Integer
3531: Function StrLength( const S : AnsiString ) : Longint
3532: Function StrRefCount( const S : AnsiString ) : Longint
3533: Procedure StrResetLength( var S : AnsiString )
3534: Function StrCharCount( const S : AnsiString; C : AnsiChar ) : Integer
3535: Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet ) : Integer
3536: Function StrStrCount( const S, SubS : AnsiString ) : Integer
3537: Function StrCompare( const S1, S2 : AnsiString ) : Integer
3538: Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer ) : Integer
3539: //Function StrFillChar( const C : AnsiChar; Count : Integer ) : AnsiString;
3540: Function StrFillChar1( const C : Char; Count : Integer ) : AnsiString
3541: Function StrFillChar( const C : Char; Count : Integer ) : string';
3542: Function IntFillChar( const I : Integer; Count : Integer ) : string') );
3543: Function ByteFillChar( const B : Byte; Count : Integer ) : string') );
3544: Function ArrFillChar( const AC : Char; Count : Integer ) : TCharArray';
3545: Function ArrByteFillChar( const AB : Char; Count : Integer ) : TByteArray;
3546: Function StrFind( const Substr, S : AnsiString; const Index : Integer ) : Integer
3547: //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString ) : Boolean
3548: Function StrIndex( const S : AnsiString; const List : array of AnsiString ) : Integer
3549: Function StrILastPos( const SubStr, S : AnsiString ) : Integer
3550: Function StrIPos( const SubStr, S : AnsiString ) : Integer
3551: Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString ) : Boolean
3552: Function StrLastPos( const SubStr, S : AnsiString ) : Integer
3553: Function StrMatch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3554: Function StrMatches( const Substr, S : AnsiString; const Index : Integer ) : Boolean
3555: Function StrNIPos( const S, SubStr : AnsiString; N : Integer ) : Integer
3556: Function StrNPos( const S, SubStr : AnsiString; N : Integer ) : Integer
3557: Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString ) : Integer
3558: Function StrSearch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3559: //Function StrAfter( const SubStr, S : AnsiString ) : AnsiString
3560: //Function StrBefore( const SubStr, S : AnsiString ) : AnsiString
3561: Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar ) : AnsiString
3562: Function StrChopRight( const S : AnsiString; N : Integer ) : AnsiString
3563: Function StrLeft( const S : AnsiString; Count : Integer ) : AnsiString
3564: Function StrMid( const S : AnsiString; Start, Count : Integer ) : AnsiString
3565: Function StrRestOf( const S : AnsiString; N : Integer ) : AnsiString
3566: Function StrRight( const S : AnsiString; Count : Integer ) : AnsiString
3567: Function CharEqualNoCase( const C1, C2 : AnsiChar ) : Boolean
3568: Function CharIsAlpha( const C : AnsiChar ) : Boolean
3569: Function CharIsAlphaNum( const C : AnsiChar ) : Boolean
3570: Function CharIsBlank( const C : AnsiChar ) : Boolean
3571: Function CharIsControl( const C : AnsiChar ) : Boolean
3572: Function CharIsDelete( const C : AnsiChar ) : Boolean
3573: Function CharIsDigit( const C : AnsiChar ) : Boolean
3574: Function CharIsLower( const C : AnsiChar ) : Boolean
3575: Function CharIsNumberChar( const C : AnsiChar ) : Boolean
3576: Function CharIsPrintable( const C : AnsiChar ) : Boolean
3577: Function CharIsPunctuation( const C : AnsiChar ) : Boolean
3578: Function CharIsReturn( const C : AnsiChar ) : Boolean
3579: Function CharIsSpace( const C : AnsiChar ) : Boolean
3580: Function CharIsUpper( const C : AnsiChar ) : Boolean
3581: Function CharIsWhiteSpace( const C : AnsiChar ) : Boolean

```

```

3582: Function CharType( const C : AnsiChar ) : Word
3583: Function CharHex( const C : AnsiChar ) : Byte
3584: Function CharLower( const C : AnsiChar ) : AnsiChar
3585: Function CharUpper( const C : AnsiChar ) : AnsiChar
3586: Function CharToggleCase( const C : AnsiChar ) : AnsiChar
3587: Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer ) : Integer
3588: Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer ) : Integer
3589: Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer ) : Integer
3590: Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar ) : Integer
3591: Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean )
3592: Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean )
3593: Function StringsToStr( const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool ):AnsiString;
3594: Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean )
3595: Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean )
3596: Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean )
3597: Function AddStringToStrings(const S:AnsiString;Strings:TStrings; const Unique:Boolean):Boolean
3598: Function BooleanToStr( B : Boolean ) : AnsiString
3599: Function FileToString( const FileName : AnsiString ) : AnsiString
3600: Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean )
3601: Procedure StringToFile2( const S, FileName : string );
3602: Function StrToken( var S : AnsiString; Separator : Ansichar ) : AnsiString
3603: Procedure StrTokens( const S : AnsiString; const List : TStrings )
3604: Procedure StrTokenToStrings( S : AnsiString; Separator : Ansichar; const List : TStrings )
3605: //Function StrWord( var S : PAnsichar; out Word : AnsiString ) : Boolean
3606: Function StrToFloatSafe( const S : AnsiString ) : Float
3607: Function StrToIntSafe( const S : AnsiString ) : Integer
3608: Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer );
3609: Function ArrayOf( List : TStrings ) : TDynStringArray;
3610:   EJclStringError', 'EJclError
3611: function IsClass(Address: TObject): Boolean;
3612: function IsObject(Address: TObject): Boolean;
3613: // Console Utilities
3614: //function ReadKey: Char;
3615: function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3616: function JclGUIDToString(const GUID: TGUID): string;
3617: function JclStringToGUID(const S: string): TGUID;
3618: end;
3619:
3620:
3621: ****uPSI_JvDBUtil;
3622: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
3623: Function GetQueryResult( const DatabaseName, SQL : string ) : Variant
3624: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant; const AResultName : string ) : Variant
3625: //Function StrFieldDesc( Field : FLDDesc ) : string
3626: Function Var2Type( V : Variant; const VarType : Integer ) : Variant
3627: Procedure CopyRecord( DataSet : TDataSet )
3628: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string; MasterField : Word; ModOp, DelOp : RINTQual )
3629: Procedure AddMasterPassword( Table : TTable; pswd : string )
3630: Procedure PackTable( Table : TTable )
3631: Procedure PackEncryptedTable( Table : TTable; pswd : string )
3632: Function EncodeQuotes( const S : string ) : string
3633: Function Cmp( const S1, S2 : string ) : Boolean
3634: Function SubStr( const S : string; const Index : Integer; const Separator : string ) : string
3635: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string ) : string
3636: Function ReplaceString( S : string; const OldPattern, NewPattern : string ) : string
3637: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer )
3638: *****uPSI_JvBDEUtils;*****
3639: //JvBDEUtils
3640: Function CreateDbLocate : TJvLocateObject
3641: //Function CheckOpen( Status : DBIResult ) : Boolean
3642: Procedure FetchAllRecords( DataSet : TBDEDDataSet )
3643: Function TransActive( Database : TDatabase ) : Boolean
3644: Function AsyncCrySupported( Database : TDatabase ) : Boolean
3645: Function GetQuoteChar( Database : TDatabase ) : string
3646: Procedure ExecuteQuery( const DbName, QueryText : string )
3647: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string )
3648: Function FieldLogicMap( FldType : TFieldType ) : Integer
3649: Function FieldSubtypeMap( FldType : TFieldType ) : Integer Value : string; Buffer : Pointer )
3650: Function GetAliasPath( const AliasName : string ) : string
3651: Function IsDirectory( const DatabaseName : string ) : Boolean
3652: Function GetBdeDirectory : string
3653: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent ) : Boolean
3654: Function DataSetFindValue( ADataSet : TBDEDDataSet; const Value, FieldName : string ) : Boolean
3655: Function DataSetFindLike( ADataSet : TBDEDDataSet; const Value, FieldName : string ) : Boolean
3656: Function DataSetRecNo( DataSet : TDataSet ) : Longint
3657: Function DataSetRecordCount( DataSet : TDataSet ) : Longint
3658: Function DataSetPositionStr( DataSet : TDataSet ) : string
3659: Procedure DataSetShowDeleted( DataSet : TBDEDDataSet; Show : Boolean )
3660: Function CurrentRecordDeleted( DataSet : TBDEDDataSet ) : Boolean
3661: Function IsFilterApplicable( DataSet : TDataSet ) : Boolean
3662: Function IsBookmarkStable( DataSet : TBDEDDataSet ) : Boolean
3663: Procedure SetIndex( Table : TTable; const IndexFieldNames : string )
3664: Procedure RestoreIndex( Table : TTable )
3665: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const )
3666: Procedure PackTable( Table : TTable )
3667: Procedure ReindexTable( Table : TTable )

```

```

3668: Procedure BdeFlushBuffers
3669: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer ) : Pointer
3670: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string )
3671: Procedure DbNotSupported
3672: Procedure ExportDataSet( Source : TBDEDDataSet; DestTable : TTable; TableType : TTableType; const
   AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint )
3673: Procedure ExportDataSetEx( Source : TBDEDDataSet; DestTable : TTable; TableType : TTableType; const
   AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter:Char;MaxRecordCount:Longint );
3674: Procedure
   ImportDataSet(Source:TBDEDDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3675: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string );
3676: *****uPSI_JvDateUtil;
3677: function CurrentYear: Word;
3678: function IsLeapYear(AYear: Integer): Boolean;
3679: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3680: function FirstDayOfPrevMonth: TDateTime;
3681: function LastDayOfPrevMonth: TDateTime;
3682: function FirstDayOfNextMonth: TDateTime;
3683: function ExtractDay(ADate: TDateTime): Word;
3684: function ExtractMonth(ADate: TDateTime): Word;
3685: function ExtractYear(ADate: TDateTime): Word;
3686: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3687: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3688: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3689: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3690: function ValidDate(ADate: TDateTime): Boolean;
3691: procedure DateDiff(Datel, Date2: TDateTime; var Days, Months, Years: Word);
3692: function MonthsBetween(Datel, Date2: TDateTime): Double;
3693: function DaysInPeriod(Datel, Date2: TDateTime): Longint;
3694: { Count days between Datel and Date2 + 1, so if Datel = Date2 result = 1 }
3695: function DaysBetween(Datel, Date2: TDateTime): Longint;
3696: { The same as previous but if Date2 < Datel result = 0 }
3697: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3698: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3699: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3700: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3701: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3702: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3703: { String to date conversions }
3704: function GetDateOrder(const DateFormat: string): TDateOrder;
3705: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3706: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3707: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3708: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3709: function DefDateFormat(FourDigitYear: Boolean): string;
3710: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3711: -----
3712: ***** JvUtils*****
3713: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3714: function GetWordOnPos(const S: string; const P: Integer): string;
3715: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3716: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3717: { SubStr returns substring from string, S, separated with Separator string}
3718: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3719: { SubStrEnd same to previous function but Index numerated from the end of string }
3720: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3721: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3722: function SubWord(P: PChar; var P2: PChar): string;
3723: { NumberByWord returns the text representation of
3724:   the number, N, in normal russian language. Was typed from Monitor magazine }
3725: function NumberByWord(const N: Longint): string;
3726: // function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3727: //the symbol Pos is pointed. Lines separated with #13 symbol }
3728: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3729: { GetXYByPos is same to previous function, but returns X position in line too}
3730: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3731: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3732: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3733: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3734: function ConcatSep(const S, S2, Separator: string): string;
3735: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3736: function ConcatLeftSep(const S, S2, Separator: string): string;
3737: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3738: function MinimizeString(const S: string; const MaxLen: Integer): string;
3739: { Next 4 function for russian chars transliterating.
3740:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3741: procedure Dos2Win(var S: string);
3742: procedure Win2Dos(var S: string);
3743: function Dos2WinRes(const S: string): string;
3744: function Win2DOSRes(const S: string): string;
3745: function Win2Koi(const S: string): string;
3746: { Spaces returns string consists on N space chars }
3747: function Spaces(const N: Integer): string;
3748: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3749: function AddSpaces(const S: string; const N: Integer): string;
3750: { function LastDate for russian users only } { returns date relative to current date: '' }
3751: function LastDate(const Dat: TDateTime): string;
3752: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3753: function CurrencyToStr(const Cur: currency): string;

```

```

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

```

```

3841: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3842: function AnsiStrICmp(S1, S2: PChar): Integer;
3843: {$ENDIF}
3844: function Var2Type(V: Variant; const VarType: Integer): Variant;
3845: function VarToInt(V: Variant): Integer;
3846: function VarToFloat(V: Variant): Double;
3847: { following functions are not documented because they are don't work properly , so don't use them }
3848: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3849: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3850: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3851: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3852: function GetParameter: string;
3853: function GetLongFileName(FileName: string): string;
3854: {* from FileCtrl}
3855: function DirectoryExists(const Name: string): Boolean;
3856: procedure ForceDirectories(Dir: string);
3857: {# from FileCtrl}
3858: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3859: function GetComputerID: string;
3860: function GetComputerName: string;
3861: {**** string routines }
3862: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
  same Index.Also see RAUtlisW.ReplaceSokr function }
3863: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3864: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
  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 }
3866: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3867: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3868: function CountOfLines(const S: string): Integer;
3869: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3870: procedure DeleteEmptyLines(Ss: TStrings);
3871: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
  Note: If strings SQL alreay contains where-statement, it must be started on begining of any line }
3873: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3874: {**** files routines - }
3875: { ResSaveToFile save resource named as Name with Typ type into file FileName.
  Resource can be compressed using MS Compress program}
3877: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3878: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool;
3879: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3880: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3881: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3882: { IniReadSection read section, Section, from ini-file,
  IniFileName, into strings, Ss.This function reads ALL strings from specified section.
  Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
3885: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3886: { LoadTextFile load text file, FileName, into string }
3887: function LoadTextFile(const FileName: TFileName): string;
3888: procedure SaveTextfile(const FileName: TFileName; const Source: string);
3889: { ReadFolder reads files list from disk folder, Folder, that are equal Mask, into strings, FileList}
3890: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3891: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3892: {$IFDEF COMPILER3_UP}
3893: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3894: function TargetFileName(const FileName: TFileName): TFileName;
3895: { return filename ShortCut linked to }
3896: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3897: {$ENDIF COMPILER3_UP}
3898: {**** Graphic routines - }
3899: { LoadIconToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3900: procedure LoadIconToImage(ALarge, ASmall: TImageList; const NameRes: string);
3901: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3902: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3903: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3904: function RATextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: string; const CalcHeight: Boolean): Integer;
3905: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3906: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3907: { Cinema draws some visual effect }
3908: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3909: { Roughed fills rect with special 3D pattern }
3910: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3911: { 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 }
3912: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3913: { TextWidth calculate text width for writing using standard desktop font }
3914: function TextWidth(AStr: string): Integer;
3915: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3916: function DefineCursor(Identifier: PChar): TCursor;
3917: {**** other routines - }
3918: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3919: function FindFormByClass(FormClass: TFormClass): TForm;
3920: function FindFormByName(ClassName: string): TForm;
3921: { FindByTag returns the control with specified class, ComponentClass, from WinControl.Controls property,
  having Tag property value, equaled to Tag parameter }
3922: { function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
  ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3924: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3925: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3926: { RBTAG searches WinControl.Controls for checked RadioButton and returns its Tag property value }

```

```

3927: function RBTag(Parent: TWinControl): Integer;
3928: { AppMinimized returns True, if Application is minimized }
3929: function AppMinimized: Boolean;
3930: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3931:   if Caption parameter = '', it replaced with Application.Title }
3932: function MessageBoxJ(const Msg: string; Caption: string; const Flags: Integer): Integer;
3933: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3934:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3935: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3936:   Buttons: TMsgDlgButtons; DefButton: TMsgDlgBtn; HelpContext: Integer; Control: TWinControl): Integer;
3937: { Delay stop program execution to MSec msec }
3938: procedure Delay(MSec: Longword);
3939: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3940: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3941: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3942: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3943: function PanelBorder(Panel: TCustomPanel): Integer;
3944: function Pixels(Control: TControl; APixels: Integer): Integer;
3945: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3946: procedure Error(const Msg: string);
3947: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3948:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3949: { ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>' }
3950: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3951:   const HideSelColor: Boolean): string;
3952: function ItemHtWidth(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3953:   const HideSelColor: Boolean): Integer;
3954: function ItemHtPlain(const Text: string): string;
3955: { ClearList - clears list of TObject }
3956: procedure ClearList(List: TList);
3957: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
3958: procedure ClipboardToMemStream(MemStream: TMemoryStream; const Format: Word);
3959: { RTTI support }
3960: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3961: function GetPropStr(Obj: TObject; const PropName: string): string;
3962: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3963: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3964: procedure PrepareIniSection(SS: TStringList);
3965: { following functions are not documented because they are don't work properly, so don't use them }
3966: {$IFDEF COMPILER2}
3967: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3968: {$ENDIF}
3969:
3970: procedure SIRegister_JvBoxProcs(CL: TPPascalCompiler);
3971: begin
3972: Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3973: Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3974: Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
3975: Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3976: Procedure BoxMoveSelected( List : TWinControl; Items : TStringList)
3977: Procedure BoxSetItem( List : TWinControl; Index : Integer)
3978: Function BoxGetFirstSelection( List : TWinControl ) : Integer
3979: Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer ) : Boolean
3980: end;
3981:
3982: procedure SIRegister_JvCsvParse(CL: TPPascalCompiler);
3983: begin
3984: Const ('MaxInitStrNum','LongInt'( 9 );
3985: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
: array of AnsiString; MaxSplit : Integer ) : Integer
3986: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
string; MaxSplit : Integer ) : Integer
3987: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3988: Function JvAnsiStrStrip( S : AnsiString ) : AnsiString
3989: Function JvStrStrip( S : string ) : string
3990: Function GetString( var Source : AnsiString; const Separator : AnsiString ) : AnsiString
3991: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar ) : AnsiString
3992: Function BuildPathName( const PathName, FileName : AnsiString ) : AnsiString
3993: Function StrEatWhiteSpace( const S : string ) : string
3994: Function HexToAscii( const S : AnsiString ) : AnsiString
3995: Function AsciiToHex( const S : AnsiString ) : AnsiString
3996: Function StripQuotes( const S1 : AnsiString ) : AnsiString
3997: Function ValidNumericLiteral( S1 : PAnsiChar ) : Boolean
3998: Function ValidIntLiteral( S1 : PAnsiChar ) : Boolean
3999: Function ValidHexLiteral( S1 : PAnsiChar ) : Boolean
4000: Function HexPCharToInt( S1 : PAnsiChar ) : Integer
4001: Function ValidStringLiteral( S1 : PAnsiChar ) : Boolean
4002: Function StripPCharQuotes( S1 : PAnsiChar ) : AnsiString
4003: Function JvValidIdentifierAnsi( S1 : PAnsiChar ) : Boolean
4004: Function JvValidIdentifier( S1 : string ) : Boolean
4005: Function JvEndChar( X : AnsiChar ) : Boolean
4006: Procedure JvGetToken( S1, S2 : PAnsiChar )
4007: Function IsExpressionKeyword( S1 : PAnsiChar ) : Boolean
4008: Function IsKeyword( S1 : PAnsiChar ) : Boolean
4009: Function JvValidVarReference( S1 : PAnsiChar ) : Boolean
4010: Function GetParenthesis( S1, S2 : PAnsiChar ) : Boolean
4011: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar )

```

```

4012: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
4013: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
4014: Function GetTokenCount : Integer;
4015: Procedure ResetTokenCount;
4016: end;
4017;
4018: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPPascalCompiler);
4019: begin
4020:   SIRegister_TJvQueryParamsDialog(CL);
4021:   Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean;
4022: end;
4023;
4024: ***** JvStringUtil / JvStringUtilities *****
4025: function FindNotBlankCharPos(const S: string): Integer;
4026: function AnsiChangeCase(const S: string): string;
4027: function GetWordOnPos(const S: string; const P: Integer): string;
4028: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4029: function Cmp(const S1, S2: string): Boolean;
4030: { Spaces returns string consists on N space chars }
4031: function Spaces(const N: Integer): string;
4032: { HasChar returns True, if char, Ch, contains in string, S }
4033: function HasChar(const Ch: Char; const S: string): Boolean;
4034: function HasAnyChar(const Chars: string; const S: string): Boolean;
4035: { SubStr returns substring from string, S, separated with Separator string}
4036: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
4037: { SubStrEnd same to previous function but Index numerated from the end of string }
4038: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4039: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
4040: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
4041: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
4042: { GetXYByPos is same to previous function, but returns X position in line too}
4043: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4044: { AddSlash returns string with added slash char to Dir parameter, if needed }
4045: function AddSlash2(const Dir: TFileName): string;
4046: { AddPath returns FileName with Path, if FileName not contain any path }
4047: function AddPath(const FileName, Path: TFileName): TFileName;
4048: { ExePath returns ExtractFilePath(ParamStr(0)) }
4049: function ExePath: TFileName;
4050: function LoadTextFile(const FileName: TFileName): string;
4051: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4052: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4053: function ConcatSep(const S, S2, Separator: string): string;
4054: { FileEqvMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4055: function FileEqvMask(FileName, Mask: TFileName): Boolean;
4056: { FileEqvMasks returns True if file, FileName, is compatible with given Masks.
4057:   Masks must be separated with comma (';') }
4058: function FileEqvMasks(FileName, Masks: TFileName): Boolean;
4059: function StringEndsWith(const Str, SubStr: string): Boolean;
4060: function ExtractFilePath2(const FileName: string): string;
4061: function StrToOem(const AnsiStr: string): string;
4062: { StrToOem translates a string from the Windows character set into the OEM character set. }
4063: function OemToAnsiStr(const OemStr: string): string;
4064: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4065: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4066: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4067: function ReplaceStr(const S, Srch, Replace: string): string;
4068: { Returns string with every occurrence of Srch string replaced with Replace string. }
4069: function DelSpace(const S: string): string;
4070: { DelSpace return a string with all white spaces removed. }
4071: function DelChars(const S: string; Chr: Char): string;
4072: { DelChars return a string with all Chr characters removed. }
4073: function DelBSpace(const S: string): string;
4074: { DelBSpace trims leading spaces from the given string. }
4075: function DelESpace(const S: string): string;
4076: { DelESpace trims trailing spaces from the given string. }
4077: function DelRSpace(const S: string): string;
4078: { DelRSpace trims leading and trailing spaces from the given string. }
4079: function DelSpace1(const S: string): string;
4080: { DelSpace1 return a string with all non-single white spaces removed. }
4081: function Tab2Space(const S: string; Numb: Byte): string;
4082: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4083: function NPos(const C: string; S: string; N: Integer): Integer;
4084: { NPos searches for a N-th position of substring C in a given string. }
4085: function MakeStr(C: Char; N: Integer): string;
4086: function MS(C: Char; N: Integer): string;
4087: { MakeStr return a string of length N filled with character C. }
4088: function AddChar(C: Char; const S: string; N: Integer): string;
4089: { AddChar return a string left-padded to length N with characters C. }
4090: function AddCharR(C: Char; const S: string; N: Integer): string;
4091: { AddCharR return a string right-padded to length N with characters C. }
4092: function LeftStr(const S: string; N: Integer): string;
4093: { LeftStr return a string right-padded to length N with blanks. }
4094: function RightStr(const S: string; N: Integer): string;
4095: { RightStr return a string left-padded to length N with blanks. }
4096: function CenterStr(const S: string; Len: Integer): string;
4097: { CenterStr centers the characters in the string based upon the Len specified. }
4098: function CompStr(const S1, S2: string): Integer;
4099: { CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2, 0 if S1 = S2, or 1 if S1>S2. }
4100: function CompText(const S1, S2: string): Integer;

```

```

4101: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4102: function Copy2Symb(const S: string; Symb: Char): string;
4103: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4104: function Copy2SymbDel(var S: string; Symb: Char): string;
4105: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4106: function Copy2Space(const S: string): string;
4107: { Copy2Symb returns a substring of a string S from beginning to first white space. }
4108: function Copy2SpaceDel(var S: string): string;
4109: { Copy2SpaceDel returns a substring of a string S from begining to first
4110:   white space and removes this substring from S. }
4111: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4112: { Returns string, with the first letter of each word in uppercase,
4113:   all other letters in lowercase. Words are delimited by WordDelims. }
4114: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4115: { WordCount given a set of word delimiters, returns number of words in S. }
4116: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4117: { Given a set of word delimiters, returns start position of N'th word in S. }
4118: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4119: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TCharSet; var Pos: Integer): string;
4120: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4121: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4122:   delimiters, return the N'th word in S. }
4123: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4124: { ExtractSubstr given set of word delimiters, returns the substring from S, that started from position Pos. }
4125: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4126: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4127: function QuotedString(const S: string; Quote: Char): string;
4128: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4129: function ExtractQuotedString(const S: string; Quote: Char): string;
4130: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4131:   and reduces pairs of Quote characters within the quoted string to a single character. }
4132: function FindPart(const HelpWilds, InputStr: string): Integer;
4133: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4134: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4135: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4136: function XorString(const Key, Src: ShortString): ShortString;
4137: function XorEncode(const Key, Source: string): string;
4138: function XorDecode(const Key, Source: string): string;
4139: { ** Command line routines ** }
4140: {$IFDEF COMPILER4_UP}
4141: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet; IgnoreCase: Boolean): Boolean;
4142: {$ENDIF}
4143: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4144: { ** Numeric string handling routines ** }
4145: function Numb2USA(const S: string): string;
4146: { Numb2USA converts numeric string S to USA-format. }
4147: function Dec2Hex(N: Longint; A: Byte): string;
4148: function D2H(N: Longint; A: Byte): string;
4149: { Dec2Hex converts the given value to a hexadecimal string representation
4150:   with the minimum number of digits (A) specified. }
4151: function Hex2Dec(const S: string): Longint;
4152: function H2D(const S: string): Longint;
4153: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4154: function Dec2Numb(N: Longint; A, B: Byte): string;
4155: { Dec2Numb converts the given value to a string representation with the
4156:   base equal to B and with the minimum number of digits (A) specified. }
4157: function Numb2Dec(S: string; B: Byte): Longint;
4158: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4159: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4160: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4161: function IntToRoman(Value: Longint): string;
4162: { IntToRoman converts the given value to a roman numeric string representation. }
4163: function RomanToInt(const S: string): Longint;
4164: { RomanToInt converts the given string to an integer value. If the string
4165:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4166: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4167: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4168: ***** JvFileUtil*****;
4169: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4170: procedure CopyFileEx(const FileName, DestName: string; OverwriteReadOnly, ShellDialog: Boolean; ProgressControl: TControl);
4171: procedure MoveFile(const FileName, DestName: TFileName);
4172: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4173: {$IFDEF COMPILER4_UP}
4174: function GetFileSize(const FileName: string): Int64;
4175: {$ELSE}
4176: function GetFileSize(const FileName: string): Longint;
4177: {$ENDIF}
4178: function FileDateTime(const FileName: string): TDateTime;
4179: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4180: function DeleteFiles(const FileMask: string): Boolean;
4181: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4182: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4183: function NormalDir(const DirName: string): string;
4184: function RemoveBackSlash(const DirName: string): string;
4185: function ValidFileName(const FileName: string): Boolean;
4186: function DirExists(Name: string): Boolean;
4187: procedure ForceDirectories(Dir: string);
4188: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;

```

```

4189: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4190: {$IFDEF COMPILER4_UP}
4191: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4192: {$ENDIF}
4193: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4194: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4195: {$IFDEF COMPILER4_UP}
4196: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4197: {$ENDIF}
4198: function GetTempDir: string;
4199: function GetWindowsDir: string;
4200: function GetSystemDir: string;
4201: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4202: {$IFDEF WIN32}
4203: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4204: function ShortToLongFileName(const ShortName: string): string;
4205: function ShortToLongPath(const ShortName: string): string;
4206: function LongToShortFileName(const LongName: string): string;
4207: function LongToShortPath(const LongName: string): string;
4208: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4209: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4210: {$ENDIF WIN32}
4211: {$IFNDEF COMPILER3_UP}
4212: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4213: {$ENDIF}
4214: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext ) : TJvCalculatorForm;
4215: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode ) : TWinControl;
4216: Function CreatePopupCalculator( AOwner : TComponent ) : TWinControl;
4217: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean );
4218:
4219: //*****procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4220: Procedure VariantClear( var V : Variant );
4221: Procedure VariantArrayRedim( var V : Variant; High : Integer );
4222: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer );
4223: Procedure VariantCpy( const src : Variant; var dst : Variant );
4224: Procedure VariantAdd( const src : Variant; var dst : Variant );
4225: Procedure VariantSub( const src : Variant; var dst : Variant );
4226: Procedure VariantMul( const src : Variant; var dst : Variant );
4227: Procedure VariantDiv( const src : Variant; var dst : Variant );
4228: Procedure VariantMod( const src : Variant; var dst : Variant );
4229: Procedure VariantAnd( const src : Variant; var dst : Variant );
4230: Procedure VariantOr( const src : Variant; var dst : Variant );
4231: Procedure VariantXor( const src : Variant; var dst : Variant );
4232: Procedure VariantShl( const src : Variant; var dst : Variant );
4233: Procedure VariantShr( const src : Variant; var dst : Variant );
4234: Function VariantAdd2( const V1 : Variant; const V2 : Variant ) : Variant;
4235: Function VariantSub2( const V1 : Variant; const V2 : Variant ) : Variant;
4236: Function VariantMul2( const V1 : Variant; const V2 : Variant ) : Variant;
4237: Function VariantDiv2( const V1 : Variant; const V2 : Variant ) : Variant;
4238: Function VariantMod2( const V1 : Variant; const V2 : Variant ) : Variant;
4239: Function VariantAnd2( const V1 : Variant; const V2 : Variant ) : Variant;
4240: Function VariantOr2( const V1 : Variant; const V2 : Variant ) : Variant;
4241: Function VariantXor2( const V1 : Variant; const V2 : Variant ) : Variant;
4242: Function VariantShl2( const V1 : Variant; const V2 : Variant ) : Variant;
4243: Function VariantShr2( const V1 : Variant; const V2 : Variant ) : Variant;
4244: Function VariantNot( const V1 : Variant ) : Variant;
4245: Function VariantNeg( const V1 : Variant ) : Variant;
4246: Function VariantGetElement( const V : Variant; il : integer ) : Variant;
4247: Function VariantGetElement1( const V : Variant; il, i2 : integer ) : Variant;
4248: Function VariantGetElement2( const V : Variant; il, i2, i3 : integer ) : Variant;
4249: Function VariantGetElement3( const V : Variant; il, i2, i3, i4 : integer ) : Variant;
4250: Function VariantGetElement4( const V : Variant; il, i2, i3, i4, i5 : integer ) : Variant;
4251: Procedure VariantPutElement( var V : Variant; const data : Variant; il : integer );
4252: Procedure VariantPutElement1( var V : Variant; const data : Variant; il, i2 : integer );
4253: Procedure VariantPutElement2( var V : Variant; const data : Variant; il, i2, i3 : integer );
4254: Procedure VariantPutElement3( var V : Variant; const data : Variant; il, i2, i3, i4 : integer );
4255: Procedure VariantPutElement4( var V : Variant; const data : Variant; il, i2, i3, i4, i5 : integer );
4256: end;
4257:
4258: *****unit uPSI_JvgUtils;*****
4259: function IsEven(I: Integer): Boolean;
4260: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4261: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4262: procedure SwapInt(var I1, I2: Integer);
4263: function Spaces(Count: Integer): string;
4264: function DupStr(const Str: string; Count: Integer): string;
4265: function DupChar(C: Char; Count: Integer): string;
4266: procedure Msg(const AMsg: string);
4267: function RectW(R: TRect): Integer;
4268: function RectH(R: TRect): Integer;
4269: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4270: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4271: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4272: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4273:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4274: procedure DrawTextInRect(DC: HDC; R: TRect; const Text: string; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4275: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4276:   Style: TglTextStyle; ADelineted, ASupress3D: Boolean; FontColor, DelinColor, HighlightColor, ShadowColor: TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);

```

```

4277: procedure DrawBox(DC:HDC; var R:TRect; Style:TGroupBoxStyle; BackgrColor: Longint; ATransparent: Boolean);
4278: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4279:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint; ATransparent: Boolean): TRect;
4280: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient; PenStyle, PenWidth: Integer);
4281: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4282: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4283:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y_in_rect!
4284:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4285:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4286: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4287:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y_in_rect!
4288:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4289:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4290: procedure BringParentWindowToTop(Wnd: TWinControl);
4291: function GetParentForm(Control: TControl): TForm;
4292: procedure GetWindowImageFrom(Control: TWinControl; X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC);
4293: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4294: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4295: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4296: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4297: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4298: function CalcMathString(AExpression: string): Single;
4299: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4300: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4301: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4302: procedure TypeStringOnKeyboard(const S: string);
4303: function NextStringGridCell(Grid: TStringGrid): Boolean;
4304: procedure DrawTextExtAligned(Canvas:TCanvas;const
4305:   Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4306: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4307: function ComponentToString(Component: TComponent): string;
4308: procedure StringToComponent(Component: TComponent; const Value: string);
4309: function PlayWaveResource(const ResName: string): Boolean;
4310: function UserName: string;
4311: function ComputerName: string;
4312: function CreateIniFileName: string;
4313: function ExpandString(const Str: string; Len: Integer): string;
4314: function Transliterate(const Str: string; RusToLat: Boolean): string;
4315: function IsSmallFonts: Boolean;
4316: function SystemColorDepth: Integer;
4317: function GetFileTypeJ(const FileName: string): TglFileType;
4318: Function GetFileType( hfile : THandle ) : DWORD';
4319: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4320: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4321:
4322: { **** Utility routines of unit classes }
4323: function LineStart(Buffer, BufPos: PChar): PChar;
4324: function ExtractStrings(Separators,WhiteSpace: TSysCharSet; Content: PChar: '+'
4325:   'Strings: TStrings): Integer
4326: Function TestStreamFormat( Stream : TStream ) : TStreamOriginalFormat
4327: Procedure RegisterClass( AClass : TPersistentClass )
4328: Procedure RegisterClasses( AClasses : array of TPersistentClass )
4329: Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string )
4330: Procedure UnRegisterClass( AClass : TPersistentClass )
4331: Procedure UnRegisterClasses( AClasses : array of TPersistentClass )
4332: Procedure UnRegisterModuleClasses( Module : HMODULE )
4333: Function FindGlobalComponent( const Name : string ) : TComponent
4334: Function IsUniqueGlobalComponentName( const Name : string ) : Boolean
4335: Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass ) : Boolean
4336: Function InitComponentRes( const ResName : string; Instance : TComponent ) : Boolean
4337: Function ReadComponentRes( const ResName : string; Instance : TComponent ) : TComponent
4338: Function ReadComponentResEx( HInstance : THandle; const ResName : string ) : TComponent
4339: Function ReadComponentResFile( const FileName : string; Instance : TComponent ) : TComponent
4340: Procedure WriteComponentResFile( const FileName : string; Instance : TComponent )
4341: Procedure GlobalFixupReferences
4342: Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings )
4343: Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4344: Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string )
4345: Procedure RemoveFixupReferences( Root : TComponent; const RootName : string )
4346: Procedure RemoveFixups( Instance : TPersistent )
4347: Function FindNestedComponent( Root : TComponent; const NamePath : string ) : TComponent
4348: Procedure BeginGlobalLoading
4349: Procedure NotifyGlobalLoading
4350: Procedure EndGlobalLoading
4351: Function GetUltimateOwner1( ACollection : TCollection ) : TPersistent;
4352: Function GetUltimateOwner( APersistent : TPersistent ) : TPersistent;
4353: // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4354: //Function MakeObjectInstance( Method : TWndMethod ) : Pointer
4355: Procedure FreeObjectInstance( ObjectInstance : Pointer )
4356: // Function AllocateHWnd( Method : TWndMethod ) : HWND
4357: Procedure DeAllocateHWnd( Wnd : HWND )
4358: Function AncestorisValid( Ancestor : TPersistent; Root, RootAncestor : TComponent ) : Boolean
4359: *****unit uPSI_SqlTimSt and DB;*****
4360: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQSLTimeStamp : TSQLTimeStamp );
4361: Function VarSQLTimeStampCreate3: Variant;
4362: Function VarSQLTimeStampCreate2( const AValue : string ) : Variant;
4363: Function VarSQLTimeStampCreate1( const AValue : TDateTime ) : Variant;
4364: Function VarSQLTimeStampCreate( const ASQSLTimeStamp : TSQLTimeStamp ) : Variant;

```

```

4365: Function VarSQLTimeStamp : TVarType
4366: Function VarIsSQLTimeStamp( const aValue : Variant ) : Boolean;
4367: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLOutputTimestamp ) : TSQLOutputTimestamp //beta
4368: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLOutputTimestamp ) : TSQLOutputTimestamp //beta
4369: Function VarToSQLTimeStamp( const aValue : Variant ) : TSQLOutputTimestamp
4370: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLOutputTimestamp ) : string
4371: Function SQLDayOfWeek( const DateTime : TSQLOutputTimestamp ) : integer
4372: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime ) : TSQLOutputTimestamp
4373: Function SQLTimeStampToDate( const DateTime : TSQLOutputTimestamp ) : TDateTime
4374: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLOutputTimestamp ) : Boolean
4375: Function StrToSQLTimeStamp( const S : string ) : TSQLOutputTimestamp
4376: Procedure CheckSQLTimeStamp( const ASQLOutputTimestamp : TSQLOutputTimestamp )
4377: Function ExtractFieldName( const Fields : string; var Pos : Integer ) : string;
4378: Function ExtractFieldName( const Fields : WideString; var Pos : Integer ) : WideString;
4379: //Procedure RegisterFields( const FieldClasses : array of TFieldClass )
4380: Procedure DatabaseError( const Message : WideString; Component : TComponent )
4381: Procedure DatabaseErrorFmt( const Message:WideString; const Args:array of const;Component:TComponent )
4382: Procedure DisposeMem( var Buffer, Size : Integer )
4383: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer ) : Boolean
4384: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4385: Function VarTypeToDataType( VarType : Integer ) : TFieldType
4386: *****unit JvStrings;*****
4387: {template functions}
4388: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4389: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4390: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4391: function RemoveMasterBlocks(const SourceStr: string): string;
4392: function RemoveFields(const SourceStr: string): string;
4393: {http functions}
4394: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4395: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4396: {set functions}
4397: procedure SplitSet(AText: string; AList: TStringList);
4398: function JoinSet(AList: TStringList): string;
4399: function FirstOfSet(const AText: string): string;
4400: function LastOfSet(const AText: string): string;
4401: function CountOfSet(const AText: string): Integer;
4402: function SetRotateRight(const AText: string): string;
4403: function SetRotateLeft(const AText: string): string;
4404: function SetPick(const AText: string; AIIndex: Integer): string;
4405: function SetSort(const AText: string): string;
4406: function SetUnion(const Set1, Set2: string): string;
4407: function SetIntersect(const Set1, Set2: string): string;
4408: function SetExclude(const Set1, Set2: string): string;
4409: {replace any <,> etc by &lt;,&gt;}
4410: function XMLSafe(const AText: string): string;
4411: {simple hash, Result can be used in Encrypt}
4412: function Hash(const AText: string): Integer;
4413: { Base64 encode and decode a string }
4414: function B64Encode(const S: AnsiString): AnsiString;
4415: function B64Decode(const S: AnsiString): AnsiString;
4416: {Basic encryption from a Borland Example}
4417: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4418: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4419: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4420: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4421: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4422: procedure CSVToTags(Src, Dst: TStringList);
4423: // converts a csv list to a tagged string list
4424: procedure TagsTOCSV(Src, Dst: TStringList);
4425: // converts a tagged string list to a csv list
4426: // only fieldnames from the first record are scanned in the other records
4427: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4428: {selects akey=avalue from Src and returns recordset in Dst}
4429: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4430: {filters Src for akey=avalue}
4431: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4432: {orders a tagged Src list by akey}
4433: function PosStr(const FindString, SourceString: string;
4434: StartPos: Integer = 1): Integer;
4435: { PosStr searches the first occurrence of a substring FindString in a string
4436: given by SourceString with case sensitivity (upper and lower case characters
4437: are differed). This function returns the index value of the first character
4438: of a specified substring from which it occurs in a given string starting with
4439: StartPos character index. If a specified substring is not found Q_PosStr
4440: returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4441: function PosStrLast(const FindString, SourceString: string): Integer;
4442: {finds the last occurrence}
4443: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4444: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4445: { PosText searches the first occurrence of a substring FindString in a string
4446: given by SourceString without case sensitivity (upper and lower case characters are not differed). This
4447: function returns the index value of the first character of a specified substring from which it occurs in a
4448: given string starting with Start
4449: function PosTextLast(const FindString, SourceString: string): Integer;
4450: {finds the last occurrence}
4451: function NameValuesToXML(const AText: string): string;
4452: {$IFDEF MSWINDOWS}
4453: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);

```

```

4452: {$ENDIF MSWINDOWS}
4453: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4454: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4455: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4456: procedure SaveString(const AFile, AText: string);
4457: Procedure SaveStringasFile( const AFile, AText : string)
4458: function LoadStringJ(const AFile: string): string;
4459: Function LoadStringOfFile( const Afile : string) : string
4460: Procedure SaveStringToFile( const AFile, AText : string)
4461: Function LoadStringFromFile( const AFile : string) : string
4462: function HexToColor(const AText: string): TColor;
4463: function UppercaseHTMLTags(const AText: string): string;
4464: function LowercaseHTMLTags(const AText: string): string;
4465: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4466: function RelativePath(const ASrc, ADst: string): string;
4467: function GetToken(var Start: Integer; const SourceText: string): string;
4468: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4469: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4470: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4471: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4472: // parses the beginning of an attribute: space + alpha character
4473: function ParseAttribute(var Start: Integer; const SourceText: string; var AName, AValue: string): Boolean;
4474: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4475: procedure ParseAttributes(const SourceText: string; Attributes: TStringList);
4476: // parses all name=value attributes to the attributes TStringList
4477: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4478: // checks if a name="value" pair exists and returns any value
4479: function GetStrValue(const AText, AName, ADefault: string): string;
4480: // retrieves string value from a line like:
4481: // name="jan verhoeven" email="jani dott verhoeven att wxs dott nl"
4482: // returns ADefault when not found
4483: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4484: // same for a color
4485: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4486: // same for an Integer
4487: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4488: // same for a float
4489: function GetBoolValue(const AText, AName: string): Boolean;
4490: // same for Boolean but without default
4491: function GetValue(const AText, AName: string): string;
4492: //retrieves string value from a line like: name="jan verhoeven" email="jani verhoeven att wxs dott nl"
4493: procedure SetValue(var AText: string; const AName, AValue: string);
4494: // sets a string value in a line
4495: procedure DeleteValue(var AText: string; const AName: string);
4496: // deletes a Name="value" pair from AText
4497: procedure GetNames(AText: string; AList: TStringList);
4498: // get a list of names from a string with name="value" pairs
4499: function GetHTMLColor(AColor: TColor): string;
4500: // converts a color value to the HTML hex value
4501: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4502: // finds a string backward case sensitive
4503: function BackPostText(Start: Integer; const FindString, SourceString: string): Integer;
4504: // finds a string backward case insensitive
4505: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string);
4506: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4507: // finds a text range, e.g. <TD>....</TD> case sensitive
4508: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string);
4509: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4510: // finds a text range, e.g. <TD>....</td> case insensitive
4511: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string);
4512: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4513: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4514: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string);
4515: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4516: // finds a text range backward, e.g. <TD>....</td> case insensitive
4517: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4518: var RangeEnd: Integer): Boolean;
4519: // finds a HTML or XML tag: <....>
4520: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string);
4521: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4522: // finds the innerText between opening and closing tags
4523: function Easter(NYear: Integer): TDateTime;
4524: // returns the easter date of a year.
4525: function GetWeekNumber(Today: TDateTime): string;
4526: //gets a datecode. Returns year and weeknumber in format: YYWW
4527: function ParseNumber(const S: string): Integer;
4528: // parse number returns the last position, starting from 1
4529: function ParseDate(const S: string): Integer;
4530: // parse a SQL style date string from positions 1,
4531: // starts and ends with #
4532:
4533: *****unit JvJCLUtils;*****
4534:
4535: function VarIsInt(Value: Variant): Boolean;
4536: // VarIsInt returns VarIsOrdinal-[varBoolean]
4537: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4538: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4539: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4540: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;

```

```

4541: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4542: function GetWordOnPos(const S: string; const P: Integer): string;
4543: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4544: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4545: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4546: { GetWordOnPosEx working like GetWordOnPos function, but
4547:   also returns Word position in iBeg, iEnd variables }
4548: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4549: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4550: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4551: function GetNextWordPosExW(const Text: WideString; StartIndex: Integer; var iBeg, iEnd: Integer): WideString;
4552: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4553: { GetEndPosCaret returns the caret position of the last char. For the position
4554:   after the last char of Text you must add 1 to the returned X value. }
4555: procedure GetEndPosCaretW(const Text: WideString; CaretX, CaretY: Integer; var X, Y: Integer);
4556: { GetEndPosCaret returns the caret position of the last char. For the position
4557:   after the last char of Text you must add 1 to the returned X value. }
4558: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4559: function SubStrBySeparator(const S:string;const Index:Integer;const
4560: Separator:string;StartIndex:Int=1):string;
4560: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
4561: Separator:WideString;StartIndex:Int:WideString;
4561: { SubStrEnd same to previous function but Index numerated from the end of string }
4562: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4563: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4564: function SubWord(P: PChar; var P2: PChar): string;
4565: function CurrencyByWord(Value: Currency): string;
4566: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4567: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4568: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4569: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4570: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4571: { ReplaceString searches for all substrings, OldPattern,
4572:   in a string, S, and replaces them with NewPattern }
4573: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4574: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
4575: WideString;StartIndex:Integer=1):WideString;
4575: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4576: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
4576: SUPPORTS_INLINE}
4577: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4578: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
4578: SUPPORTS_INLINE}
4579:
4580: { Next 4 function for russian chars transliterating.
4581:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4582: procedure Dos2Win(var S: AnsiString);
4583: procedure Win2Dos(var S: AnsiString);
4584: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4585: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4586: function Win2Koi(const S: AnsiString): AnsiString;
4587: { FillString fills the string Buffer with Count Chars }
4588: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4589: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4590: { MoveString copies Count Chars from Source to Dest }
4591: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4592: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4593: DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4594: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4595: procedure FillWideChar(var Buffer: string; Count: Integer; const Value: WideChar);
4596: { MoveWideChar copies Count WideChars from Source to Dest }
4597: procedure MoveWideChar(const Source: string; var Dest: string;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
4597: SUPPORTS_INLINE}
4598: { FillNativeChar fills Buffer with Count NativeChars }
4599: procedure FillNativeChar(var Buffer: string; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
4599: SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4600: { MoveWideChar copies Count WideChars from Source to Dest }
4601: procedure MoveNativeChar(const Source: string; var Dest: string; Count: Integer); // D2009 internal error {$IFDEF
4601: SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4602: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4603: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4604: { Spaces returns string consists on N space chars }
4605: function Spaces(const N: Integer): string;
4606: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4607: function AddSpaces(const S: string; const N: Integer): string;
4608: function SpacesW(const N: Integer): WideString;
4609: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4610: { function LastDateRUS for russian users only }
4611: { returns date relative to current date: 'äà àíÿ íàçàää' }
4612: function LastDateRUS(const Dat: TDateTime): string;
4613: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4614: function CurrencyToStr(const Cur: Currency): string;
4615: { HasChar returns True, if Char, Ch, contains in string, S }
4616: function HasChar(const Ch: Char; const S: string): Boolean;
4617: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4618: function HasAnyChar(const Chars: string; const S: string): Boolean;
4619: {$IFNDEF COMPILER12_UP}
4620: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}

```

```

4621: {$ENDIF ~COMPILER12_UP}
4622: function CharInSetW(const Ch: WideChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4623: function CountOfChar(const Ch: Char; const S: string): Integer;
4624: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4625: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4626: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4627: function StrPosW(S, SubStr: PWideChar): PWideChar;
4628: function StrLenW(S: PWideChar): Integer;
4629: function TrimW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4630: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4631: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4632: TPixelFormat', '(pfDevice, pflbit, pf4bit, pf8bit, pf24bit )
4633: TMappingMethod', '(mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4634: Function GetBitmapPixelFormat( Bitmap: TBitmap ): TPixelFormat
4635: Function GetPaletteBitmapFormat( Bitmap : TBitmap ) : TPixelFormat
4636: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4637: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4638: Procedure GrayscaleBitmap( Bitmap : TBitmap )
4639: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer ) : TStream
4640: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer )
4641: Function ScreenColorFormat : TPixelFormat
4642: Function ScreenColorCount : Integer
4643: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic )
4644: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean ) : TPoint
4645: // SIRegister_TJvGradient(CL);
4646:
4647: {***** files routines}
4648: procedure SetDelimitedText(List: TStringList; const Text: string; Delimiter: Char);
4649: const
4650: {$IFDEF MSWINDOWS}
4651: DefaultCaseSensitivity = False;
4652: {$ENDIF MSWINDOWS}
4653: {$IFDEF UNIX}
4654: DefaultCaseSensitivity = True;
4655: {$ENDIF UNIX}
4656: { GenTempFileName returns temporary file name on
4657: drive, there FileName is placed }
4658: function GenTempFileName(FileName: string): string;
4659: { GenTempFileNameExt same to previous function, but
4660: returning filename has given extension, FileExt }
4661: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4662: { ClearDir clears folder Dir }
4663: function ClearDir(const Dir: string): Boolean;
4664: { DeleteDir clears and than delete folder Dir }
4665: function DeleteDir(const Dir: string): Boolean;
4666: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4667: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4668: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4669: Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4670: function FileEquMasks(FileName, Masks: TFileName;
4671: CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4672: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4673: {$IFDEF MSWINDOWS}
4674: { LZFileExpand expand file, FileSource,
4675: into FileDest. Given file must be compressed, using MS Compress program }
4676: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4677: {$ENDIF MSWINDOWS}
4678: { FileGetInfo fills SearchRec record for specified file attributes}
4679: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4680: { HasSubFolder returns True, if folder APath contains other folders }
4681: function HasSubFolder(APath: TFileName): Boolean;
4682: { IsEmptyFolder returns True, if there are no files or
4683: folders in given folder, APath}
4684: function IsEmptyFolder(APath: TFileName): Boolean;
4685: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4686: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4687: { AddPath returns FileName with Path, if FileName not contain any path }
4688: function AddPath(const FileName, Path: TFileName): TFileName;
4689: function AddPaths(const Pathlist, Path: string): string;
4690: function ParentPath(const Path: TFileName): TFileName;
4691: function FindInPath(const FileName, PathList: string): TFileName;
4692: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4693: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4694: { HasParam returns True, if program running with specified parameter, Param }
4695: function HasParam(const Param: string): Boolean;
4696: function HasSwitch(const Param: string): Boolean;
4697: function Switch(const Param: string): string;
4698: { ExePath returns ExtractFilePath(ParamStr(0)) }
4699: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4700: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4701: //function FileTimeToDate(FT: TFileTime): TDateTime;
4702: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4703: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4704: {**** Graphic routines }
4705: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4706: function IsTTFontSelected(const DC: HDC): Boolean;

```

```

4707: function KeyPressed(VK: Integer): Boolean;
4708: Function isKeyPressed: boolean; //true if key on memo2 (shell output) is pressed
4709: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4710: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4711: {**** Color routines }
4712: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4713: function RGBToBGR(Value: Cardinal): Cardinal;
4714: //function ColorToPrettyName(Value: TColor): string;
4715: //function PrettyNameToColor(const Value: string): TColor;
4716: {**** other routines }
4717: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4718: function IntPower(Base, Exponent: Integer): Integer;
4719: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4720: function StrToBool(const S: string): Boolean;
4721: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4722: function VarToInt(V: Variant): Integer;
4723: function VarToFloat(V: Variant): Double;
4724: { following functions are not documented because they not work properly sometimes, so do not use them }
4725: // (rom) ReplaceStrings1, GetSubStr removed
4726: function GetLongFileName(const FileName: string): string;
4727: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4728: function GetParameter: string;
4729: function GetComputerID: string;
4730: function GetComputerName: string;
4731: {**** string routines }
4732: { ReplaceAllStrings searches for all substrings, Words,
4733:   in a string, S, and replaces them with Frases with the same Index. }
4734: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4735: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4736:   in the list, Words, and if founds, replaces this Word with string from another list,Frases, with the
4737:   same Index, and then update NewSelStart variable }
4738: function ReplaceStrings(const S:string;PosBeg:PosInt;Words,Frases:TStrings;var NewSelStart:Int):string;
4739: { CountOfLines calculates the lines count in a string, S,
4740:   each line must be separated from another with CrLf sequence }
4740: function CountOfLines(const S: string): Integer;
4741: { DeleteLines deletes all lines from strings which in the words, words.
4742:   The word of will be deleted from strings. }
4743: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4744: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4745:   Lines contained only spaces also deletes. }
4746: procedure DeleteEmptyLines(Ss: TStrings);
4747: { SQLAddWhere addes or modifies existing where-statement, where,
4748:   to the strings, SQL. Note: If strings SQL already contains where-statement,
4749:   it must be started on the begining of any line }
4750: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4751: {**** files routines - }
4752: {$IFDEF MSWINDOWS}
4753: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4754:   Resource can be compressed using MS Compress program}
4755: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4756: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string): Boolean;
4757: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4758: {$ENDIF MSWINDOWS}
4759: { IniReadSection read section, Section, from ini-file,
4760:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4761:   Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
4762: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4763: { LoadTextFile load text file, FileName, into string }
4764: function LoadTextFile(const FileName: TFileName): string;
4765: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4766: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, fileList}
4767: function ReadFolder(const Folder, Mask: TFileName; fileList: TStrings): Integer;
4768: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4769: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4770: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4771: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4772: function RATextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S:string;const CalcHeight:Boolean):Integer;
4773: { RATextCalcHeight calculate needed height for
4774:   correct output, using RATextOut or RATextOutEx functions }
4775: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4776: { Cinema draws some visual effect }
4777: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4778: { Roughed fills rect with special 3D pattern }
4779: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4780: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
4781:   and height, AWidth, AHeight and placed on a specified Index, Index in the source bitmap }
4782: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4783: function TextWidth(const AStr: string): Integer;
4784: { TextHeight calculate text height for writing using standard desktop font }
4785: function TextHeight(const AStr: string): Integer;
4786: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4787: procedure Error(const Msg: string);
4788: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4789:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4790: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4791: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4792:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;

```

```

4793: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4794:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4795: function ItemHtPlain(const Text: string): string;
4796: { ClearList - clears list of TObject }
4797: procedure ClearList(List: TList);
4798: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
4799: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4800: { RTTI support }
4801: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4802: function GetPropStr(Obj: TObject; const PropName: string): string;
4803: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4804: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4805: procedure PrepareIniSection(Ss: TStrings);
4806: { following functions are not documented because they are don't work properly, so don't use them }
4807: // (rom) from JvBandWindows to make it obsolete
4808: function PointL(const X, Y: Longint): TPointI; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4809: // (rom) from JvBandUtils to make it obsolete
4810: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4811: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4812: function CreateIconFromClipboard: TIcon;
4813: { begin JvIconClipboardUtils } { Icon clipboard routines }
4814: function CF_ICON: Word;
4815: procedure AssignClipboardIcon(Icon: TIcon);
4816: { Real-size icons support routines (32-bit only) }
4817: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4818: function CreateRealSizeIcon(Icon: TIcon): HICON;
4819: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4820: {end JvIconClipboardUtils }
4821: function CreateScreenCompatibleDC: HDC;
4822: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF}
4823: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4824: { begin JvRLE } // (rom) changed API for inclusion in JCL
4825: procedure RleCompressTo(InStream, OutStream: TStream);
4826: procedure RleDecompressTo(InStream, OutStream: TStream);
4827: procedure RleCompress(Stream: TStream);
4828: procedure RleDecompress(Stream: TStream);
4829: { end JvRLE } { begin JvDateUtil }
4830: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4831: function IsLeapYear(AYear: Integer): Boolean;
4832: function DaysInAMonth(const AYear, AMonth: Word): Word;
4833: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4834: function FirstDayOfPrevMonth: TDateTime;
4835: function LastDayOfPrevMonth: TDateTime;
4836: function FirstDayOfNextMonth: TDateTime;
4837: function ExtractDay(ADate: TDateTime): Word;
4838: function ExtractMonth(ADate: TDateTime): Word;
4839: function ExtractYear(ADate: TDateTime): Word;
4840: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4841: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4842: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4843: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4844: function ValidDate(ADate: TDateTime): Boolean;
4845: procedure DateDiff(Datel, Date2: TDateTime; var Days, Months, Years: Word);
4846: function MonthsBetween(Datel, Date2: TDateTime): Double;
4847: function DaysInPeriod(Datel, Date2: TDateTime): Longint;
4848: { Count days between Datel and Date2 + 1, so if Datel = Date2 result = 1 }
4849: function DaysBetween(Datel, Date2: TDateTime): Longint;
4850: { The same as previous but if Date2 < Datel result = 0 }
4851: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4852: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4853: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4854: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4855: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4856: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
4857: { String to date conversions }
4858: function GetDateOrder(const DateFormat: string): TDateOrder;
4859: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4860: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4861: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4862: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4863: {function DefDateFormat(AFourDigitYear: Boolean): string;
4864: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4865: function FormatLongDate(Value: TDateTime): string;
4866: function FormatLongDateTime(Value: TDateTime): string;
4867: { end JvDateUtil }
4868: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4869: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4870: { begin JvStrUtils } { ** Common string handling routines ** }
4871: {$IFDEF UNIX}
4872: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4873:   const ToCode, FromCode: AnsiString): Boolean;
4874: function iconvstring(const S, ToCode, FromCode: AnsiString): string;
4875: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4876: function OemStrToAnsi(const S: AnsiString): AnsiString;
4877: function AnsiStrToOem(const S: AnsiString): AnsiString;
4878: {$ENDIF UNIX}
4879: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4880: { StrToOem translates a string from the Windows character set into the OEM character set. }

```

```

4881: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4882: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4883: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4884: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4885: function ReplaceStr(const S, Srch, Replace: string): string;
4886: { Returns string with every occurrence of Srch string replaced with Replace string. }
4887: function DelSpace(const S: string): string;
4888: { DelSpace return a string with all white spaces removed. }
4889: function DelChars(const S: string; Chr: Char): string;
4890: { DelChars return a string with all Chr characters removed. }
4891: function DelBSpace(const S: string): string;
4892: { DelBSpace trims leading spaces from the given string. }
4893: function DelESpace(const S: string): string;
4894: { DelESpace trims trailing spaces from the given string. }
4895: function DelRSpace(const S: string): string;
4896: { DelRSpace trims leading and trailing spaces from the given string. }
4897: function DelSpace1(const S: string): string;
4898: { DelSpace1 return a string with all non-single white spaces removed. }
4899: function Tab2Space(const S: string; Numb: Byte): string;
4900: { Tab2Space converts any tabulation character in the given string to the
4901:   Numb spaces characters. }
4902: function NPos(const C: string; S: string; N: Integer): Integer;
4903: { NPos searches for a N-th position of substring C in a given string. }
4904: function MakeStr(C: Char; N: Integer): string; overload;
4905: {$IFNDEF COMPILER12_UP}
4906: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4907: {$ENDIF !COMPILER12_UP}
4908: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4909: { MakeStr return a string of length N filled with character C. }
4910: function AddChar(C: Char; const S: string; N: Integer): string;
4911: { AddChar return a string left-padded to length N with characters c. }
4912: function AddCharR(C: Char; const S: string; N: Integer): string;
4913: { AddCharR return a string right-padded to length N with characters C. }
4914: function LeftStr(const S: string; N: Integer): string;
4915: { LeftStr return a string right-padded to length N with blanks. }
4916: function RightStr(const S: string; N: Integer): string;
4917: { RightStr return a string left-padded to length N with blanks. }
4918: function CenterStr(const S: string; Len: Integer): string;
4919: { CenterStr centers the characters in the string based upon the Len specified. }
4920: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4921: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4922:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4923: function CompText(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4924: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4925: function Copy2Symb(const S: string; Symb: Char): string;
4926: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4927: function Copy2SymbDel(var S: string; Symb: Char): string;
4928: { Copy2SymbDel returns a substring of a string S from begining to first
4929:   character Symb and removes this substring from S. }
4930: function Copy2Space(const S: string): string;
4931: { Copy2Space returns a substring of a string S from begining to first white space. }
4932: function Copy2SpaceDel(var S: string): string;
4933: { Copy2SpaceDel returns a substring of a string S from begining to first
4934:   white space and removes this substring from S. }
4935: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4936: { Returns string, with the first letter of each word in uppercase,
4937:   all other letters in lowercase. Words are delimited by WordDelims. }
4938: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4939: { WordCount given a set of word delimiters, returns number of words in S. }
4940: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4941: { Given a set of word delimiters, returns start position of N'th word in S. }
4942: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4943: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TSysCharSet; var Pos: Integer): string;
4944: function ExtractDelimited(N: Integer; const S: string; const Delims: TSysCharSet): string;
4945: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4946:   delimiters, return the N'th word in S. }
4947: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TSysCharSet): string;
4948: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4949:   that started from position Pos. }
4950: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4951: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4952: function QuotedString(const S: string; Quote: Char): string;
4953: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4954: function ExtractQuotedString(const S: string; Quote: Char): string;
4955: { ExtractQuotedString removes the Quote characters from the beginning and
4956:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character. }
4957: function FindPart(const HelpWilds, InputStr: string): Integer;
4958: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4959: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4960: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4961: function XorString(const Key, Src: shortString): shortString;
4962: function XorEncode(const Key, Source: string): string;
4963: function XorDecode(const Key, Source: string): string;
4964: { ** Command line routines ** }
4965: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4966: { ** Numeric string handling routines ** }
4967: function Numb2USA(const S: string): string;
4968: { Numb2USA converts numeric string S to USA-format. }
4969: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}

```

```

4970: { Dec2Hex converts the given value to a hexadecimal string representation
4971:   with the minimum number of digits (A) specified. }
4972: function Hex2Dec(const S: string): Longint;
4973: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4974: function Dec2Numb(N: Int64; A, B: Byte): string;
4975: { Dec2Numb converts the given value to a string representation with the
4976:   base equal to B and with the minimum number of digits (A) specified. }
4977: function Numb2Dec(S: string; B: Byte): Int64;
4978: { Numb2Dec converts the given B-based numeric string to the corresponding
4979:   integer value. }
4980: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4981: { IntToBin converts the given value to a binary string representation
4982:   with the minimum number of digits specified. }
4983: function IntToRoman(Value: Longint): string;
4984: { IntToRoman converts the given value to a roman numeric string representation. }
4985: function RomanToInt(const S: string): Longint;
4986: { RomanToInt converts the given string to an integer value. If the string
4987:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4988: function FindNotBlankCharPos(const S: string): Integer;
4989: function FindNotBlankCharPosW(const S: WideString): Integer;
4990: function AnsiChangeCase(const S: string): string;
4991: function WideChangeCase(const S: string): string;
4992: function StartsText(const SubStr, S: string): Boolean;
4993: function EndsText(const SubStr, S: string): Boolean;
4994: function DequotedStr(const S: string; QuoteChar: Char = ''): string;
4995: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4996: {end JvStrUtils}
4997: {$IFDEF UNIX}
4998: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4999: {$ENDIF UNIX}
5000: { begin JvFileUtil }
5001: function FileDateTime(const FileName: string): TDateTime;
5002: function HasAttr(const FileName: string; Attr: Integer): Boolean;
5003: function DeleteFilesEx(const FileMasks: array of string): Boolean;
5004: function NormalDir(const DirName: string): string;
5005: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
5006: function ValidfileName(const FileName: string): Boolean;
5007: {$IFDEF MSWINDOWS}
5008: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5009: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5010: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5011: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5012: {$ENDIF MSWINDOWS}
5013: function GetWindowsDir: string;
5014: function GetSystemDir: string;
5015: function ShortToLongFileName(const ShortName: string): string;
5016: function LongToShortFileName(const LongName: string): string;
5017: function ShortToLongPath(const ShortName: string): string;
5018: function LongToShortPath(const LongName: string): string;
5019: {$IFDEF MSWINDOWS}
5020: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
5021: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
5022: {$ENDIF MSWINDOWS}
5023: { end JvFileUtil }
5024: // Works like PtInRect but includes all edges in comparision
5025: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
5026: // Works like PtInRect but excludes all edges from comparision
5027: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
5028: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5029: function IsFourDigitYear: Boolean;
5030: { moved from JvJVCLUTils }
5031: //Open an object with the shell (url or something like that)
5032: function OpenObject(const Value: string): Boolean; overload;
5033: function OpenObject(Value: PChar): Boolean; overload;
5034: {$IFDEF MSWINDOWS}
5035: //Raise the last Exception
5036: procedure RaiseLastWin32; overload;
5037: procedure RaiseLastWin32(const Text: string); overload;
5038: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
5039: // significant 32 bits of a file's binary version number. Typically, this includes the major and minor
5040: // version placed together in one 32-bit Integer. I
5041: function GetFileVersion(const AFileName: string): Cardinal;
5042: { EXTERNALSYM GetFileVersion}
5043: //Get version of Shell.dll
5044: function GetShellVersion: Cardinal;
5045: {$ENDIF MSWINDOWS}
5046: { EXTERNALSYM GetShellVersion}
5047: // CD functions on HW
5048: procedure OpenCdDrive;
5049: procedure CloseCdDrive;
5050: // returns True if Drive is accessible
5051: function DiskInDrive(Drive: Char): Boolean;
5052: {$ENDIF MSWINDOWS}
5053: //Same as linux function ;
5054: procedure PError(const Text: string);
5055: // execute a program without waiting
5056: procedure Exec(const FileName, Parameters, Directory: string);
5057: // execute a program and wait for it to finish
5058: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5059: // returns True if this is the first instance of the program that is running

```

```

5057: function FirstInstance(const ATitle: string): Boolean;
5058: // restores a window based on it's classname and Caption. Either can be left empty
5059: // to widen the search
5060: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5061: // manipulate the traybar and start button
5062: procedure HideTraybar;
5063: procedure ShowTraybar;
5064: procedure ShowStartButton(Visible: Boolean = True);
5065: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5066: procedure MonitorOn;
5067: procedure MonitorOff;
5068: procedure LowPower;
5069: // send a key to the window named AppName
5070: function SendKey(const AppName: string; Key: Char): Boolean;
5071: {$IFDEF MSWINDOWS}
5072: // returns a list of all win currently visible, the Objects property is filled with their window handle
5073: procedure GetVisibleWindows(List: TStrings);
5074: Function GetVisibleWindowsF( List : TStrings):TStrings';
5075: // associates an extension to a specific program
5076: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5077: procedure AddToRecentDocs(const FileName: string);
5078: function GetRecentDocs: TStringList;
5079: {$ENDIF MSWINDOWS}
5080: function CharIsMoney(const Ch: Char): Boolean;
5081: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5082: function IntToExtended(I: Integer): Extended;
5083: { GetChangedText works out the new text given the current cursor pos & the key pressed
5084: It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5085: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5086: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5087: //function StrIsInteger(const S: string): Boolean;
5088: function StrIsFloatMoney(const Ps: string): Boolean;
5089: function StrIsDateTime(const Ps: string): Boolean;
5090: function PreformatDateString(Ps: string): string;
5091: function BooleanToInteger(const B: Boolean): Integer;
5092: function StringToBoolean(const Ps: string): Boolean;
5093: function SafeStrToDate(const Ps: string): TDateTime;
5094: function SafeStrToDate(const Ps: string): TDateTime;
5095: function SafeStrToTime(const Ps: string): TDateTime;
5096: function StrDelete(const psSub, psMain: string): string;
5097: { returns the fractional value of pcValue}
5098: function TimeOnly(pcValue: TDateTime): TTime;
5099: { returns the integral value of pcValue }
5100: function DateOnly(pcValue: TDateTime): TDate;
5101: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5102: const { TDateTime value used to signify Null value}
5103: NullEquivalentDate: TDateTime = 0.0;
5104: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5105: // Replacement for Win32Check to avoid platform specific warnings in D6
5106: function OSCheckRetVal: Boolean;
5107: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5108: Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5109: not be forced to use FileCtrl unnecessarily }
5110: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5111: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5112: { MinimizeString truncates long string, S, and appends...'symbols,if Length of S is more than MaxLen }
5113: function MinimizeString(const S: string; const MaxLen: Integer): string;
5114: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=SW_SHOWDEFAULT);
5115: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion function and calls it, returning major and
5116: minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't be
5117: found.}
5116: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5117: {$ENDIF MSWINDOWS}
5118: procedure ResourceNotFound(ResID: PChar);
5119: function EmptyRect: TRect;
5120: function RectWidth(R: TRect): Integer;
5121: function RectHeight(R: TRect): Integer;
5122: function CompareRect(const R1, R2: TRect): Boolean;
5123: procedure RectNormalize(var R: TRect);
5124: function RectIsSquare(const R: TRect): Boolean;
5125: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5126: //If AMaxSize = -1 ,then auto calc Square's max size
5127: {$IFDEF MSWINDOWS}
5128: procedure FreeUnusedOLE;
5129: function GetWindowsVersion: string;
5130: function LoadDLL(const LibName: string): THandle;
5131: function RegisterServer(const ModuleName: string): Boolean;
5132: function UnregisterServer(const ModuleName: string): Boolean;
5133: {$ENDIF MSWINDOWS}
5134: { String routines }
5135: function GetEnvVar(const VarName: string): string;
5136: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5137: function StringToPChar(var S: string): PChar;
5138: function StrPAlloc(const S: string): PChar;
5139: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5140: function DropT(const S: string): string;
5141: { Memory routines }
5142: function AllocMemo(Size: Longint): Pointer;

```

```

5143: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5144: procedure FreeMemo(var fpBlock: Pointer);
5145: function GetMemoSize(fpBlock: Pointer): Longint;
5146: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5147: { Manipulate huge pointers routines }
5148: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5149: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5150: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5151: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5152: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5153: function WindowClassName(Wnd: THandle): string;
5154: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5155: procedure ActivateWindow(Wnd: THandle);
5156: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5157: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5158: { SetWindowTop put window to top without recreating window }
5159: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5160: procedure CenterWindow(Wnd: THandle);
5161: function MakeVariant(const Values: array of Variant): Variant;
5162: { Convert dialog units to pixels and backwards }
5163: {$IFDEF MSWINDOWS}
5164: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5165: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5166: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5167: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5168: {$ENDIF MSWINDOWS}
5169: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5170: {$IFDEF BCB}
5171: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): THandle;
5172: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
5173: {$ELSE}
5174: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5175: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5176: {$ENDIF BCB}
5177: {$IFDEF MSWINDOWS}
5178: { BrowseForFolderNative displays Browse For Folder dialog }
5179: function BrowseForFolderNative(Handle: THandle; const Title: string; var Folder: string): Boolean;
5180: {$ENDIF MSWINDOWS}
5181: procedure AntiAlias(Clip: TBitmap);
5182: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin, XFinal, YFinal: Integer);
5183: procedure CopyRectIBits(ACanvas: TCanvas; const DestRect: TRect;
5184: ABitmap: TBitmap; const SourceRect: TRect);
5185: function IsTrueType(const FontName: string): Boolean;
5186: // Removes all non-numeric characters from AValue and returns the resulting string
5187: function TextToValText(const AValue: string): string;
5188: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString ) : boolean;
5189: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings );
5190: Function ReplaceRegExpr( const ARegExpr,AInputStr,
5191: AReplaceStr:RegExprString;AUseSubstitution:boolean ):RegExprString;
5192: Function RegExprSubExpressions( const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean ) :
5193:
5194: *****unit uPSI_JvTFUtils;
5195: Function JExtractYear( ADate : TDateTime ) : Word;
5196: Function JExtractMonth( ADate : TDateTime ) : Word;
5197: Function JExtractDay( ADate : TDateTime ) : Word;
5198: Function ExtractHours( ATime : TDateTime ) : Word;
5199: Function ExtractMins( ATime : TDateTime ) : Word;
5200: Function ExtractSecs( ATime : TDateTime ) : Word;
5201: Function ExtractMSecs( ATime : TDateTime ) : Word;
5202: Function FirstOfMonth( ADate : TDateTime ) : TDateTime;
5203: Function GetDayOfNthDOW( Year, Month, DOW, N : Word ) : Word;
5204: Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer ) : Word;
5205: Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer );
5206: Procedure IncDOW( var DOW : TTFFDayOfWeek; N : Integer );
5207: Procedure IncDays( var ADate : TDateTime; N : Integer );
5208: Procedure IncWeeks( var ADate : TDateTime; N : Integer );
5209: Procedure IncMonths( var ADate : TDateTime; N : Integer );
5210: Procedure IncYears( var ADate : TDateTime; N : Integer );
5211: Function EndOfMonth( ADate : TDateTime ) : TDateTime;
5212: Function IsFirstOfMonth( ADate : TDateTime ) : Boolean;
5213: Function IsEndOfMonth( ADate : TDateTime ) : Boolean;
5214: Procedure EnsureMonth( Month : Word );
5215: Procedure EnsureDOW( DOW : Word );
5216: Function EqualDates( D1, D2 : TDateTime ) : Boolean;
5217: Function Lesser( N1, N2 : Integer ) : Integer;
5218: Function Greater( N1, N2 : Integer ) : Integer;
5219: Function GetDivLength( TotalLength, DivCount, DivNum : Integer ) : Integer;
5220: Function GetDivNum( TotalLength, DivCount, X : Integer ) : Integer;
5221: Function GetDivStart( TotalLength, DivCount, DivNum : Integer ) : Integer;
5222: Function DOWToBorl( ADOW : TTFFDayOfWeek ) : Integer;
5223: Function BorlToDOW( BorlDOW : Integer ) : TTFFDayOfWeek;
5224: Function DateToDOW( ADate : TDateTime ) : TTFFDayOfWeek;
5225: Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
5226: AFont:TFont; AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string );
5227: Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
5228: HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string );
5229: Function JRectWidth( ARect : TRect ) : Integer;
5230: Function JRectHeight( ARect : TRect ) : Integer;

```

```

5229: Function JEmptyRect : TRect
5230: Function IsClassByName( Obj : TObject; ClassName : string ) : Boolean
5231:
5232: procedure SIRегистер_MSysUtils(CL: TPSPascalCompiler);
5233: begin
5234:   Procedure HideTaskBarButton( hWindow : HWND )
5235:   Function msLoadStr( ID : Integer ) : String
5236:   Function msFormat( fmt : String; params : array of const ) : String
5237:   Function msFileExists( const FileName : String ) : Boolean
5238:   Function msIntToStr( Int : Int64 ) : String
5239:   Function msStrPas( const Str : PChar ) : String
5240:   Function msRenameFile( const OldName, NewName : String ) : Boolean
5241:   Function CutFileName( s : String ) : String
5242:   Function GetVersionInfo( var VersionString : String ) : DWORD
5243:   Function FormatTime( t : Cardinal ) : String
5244:   Function msCreateDir( const Dir : string ) : Boolean
5245:   Function SetAutoRun( NeedAutoRun : Boolean; AppName : String ) : Boolean
5246:   Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword ) : DWORD
5247:   Function msStrLen( Str : PChar ) : Integer
5248:   Function msDirectoryExists( const Directory : String ) : Boolean
5249:   Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String ) : String
5250:   Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte ) : LongBool
5251:   Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
5252:   Procedure SetEditWndProc( hWnd : HWND; ptr : TObject )
5253:   Function GetTextFromFile( Filename : String ) : string
5254:   Function IsTopMost( hWnd : HWND ) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002 );
5255:   Function msStrToIntDef( const s : String; const i : Integer ) : Integer
5256:   Function msStrToInt( s : String ) : Integer
5257:   Function GetItemText( hDlg : THandle; ID : DWORD ) : String
5258: end;
5259:
5260: procedure SIRегистер_ESBMaths2(CL: TPSPascalCompiler);
5261: begin
5262:   //TDynFloatArray', 'array of Extended
5263:   TDynLWordArray', 'array of LongWord
5264:   TDynLIntArray', 'array of LongInt
5265:   TDynFloatMatrix', 'array of TDynFloatArray
5266:   TDynLWordMatrix', 'array of TDynLWordArray
5267:   TDynLIntMatrix', 'array of TDynLIntArray
5268:   Function SquareAll( const X : TDynFloatArray ) : TDynFloatArray
5269:   Function InverseAll( const X : TDynFloatArray ) : TDynFloatArray
5270:   Function LnAll( const X : TDynFloatArray ) : TDynFloatArray
5271:   Function Log10All( const X : TDynFloatArray ) : TDynFloatArray
5272:   Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended ) : TDynFloatArray
5273:   Function AddVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5274:   Function SubVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5275:   Function MultVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5276:   Function DotProduct( const X, Y : TDynFloatArray ) : Extended
5277:   Function MNorm( const X : TDynFloatArray ) : Extended
5278:   Function MatrixIsRectangular( const X : TDynFloatMatrix ) : Boolean
5279:   Procedure MatrixDimensions( const X:TDynFloatMatrix; var Rows,Columns:LongWord; var Rectangular:Boolean );
5280:   Function MatrixIsSquare( const X : TDynFloatMatrix ) : Boolean
5281:   Function MatricesSameDimensions( const X, Y : TDynFloatMatrix ) : Boolean
5282:   Function AddMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix
5283:   Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix )
5284:   Function SubtractMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix
5285:   Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix )
5286:   Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended ) : TDynFloatMatrix
5287:   Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended );
5288:   Function MultiplyMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix;
5289:   Function TransposeMatrix( const X : TDynFloatMatrix ) : TDynFloatMatrix;
5290:   Function GrandMean( const X : TDynFloatMatrix; var N : LongWord ) : Extended
5291: end;
5292:
5293: procedure SIRегистер_ESBMaths(CL: TPSPascalCompiler);
5294: begin
5295:   'ESBMinSingle','Single').setExtended( 1.5e-45 );
5296:   'ESBMaxSingle','Single').setExtended( 3.4e+38 );
5297:   'ESBMinDouble','Double').setExtended( 5.0e-324 );
5298:   'ESBMaxDouble','Double').setExtended( 1.7e+308 );
5299:   'ESBMinExtended','Extended').setExtended( 3.6e-4951 );
5300:   'ESBMaxExtended','Extended').setExtended( 1.1e+4932 );
5301:   'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807 );
5302:   'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807 );
5303:   'ESBSqrt2','Extended').setExtended( 1.4142135623730950488 );
5304:   'ESBSqrt3','Extended').setExtended( 1.7320508075688772935 );
5305:   'ESBSqrt5','Extended').setExtended( 2.2360679774997896964 );
5306:   'ESBSqrt10','Extended').setExtended( 3.1622776601683793320 );
5307:   'ESBCbrtPi','Extended').setExtended( 1.77245385090551602729 );
5308:   'ESBCbrt2','Extended').setExtended( 1.2599210498948731648 );
5309:   'ESBCbrt3','Extended').setExtended( 1.4422495703074083823 );
5310:   'ESBCbrt10','Extended').setExtended( 2.1544346900318837219 );
5311:   'ESBCbrt100','Extended').setExtended( 4.6415888336127788924 );
5312:   'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630 );
5313:   'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440 );
5314:   'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451 );
5315:   'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928 );
5316:   'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695 );
5317:   'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147 );

```

```

5318: 'ESBe','Extended').setExtended( 2.7182818284590452354);
5319: 'ESBe2','Extended').setExtended( 7.3890560989306502272);
5320: 'ESBePi','Extended').setExtended( 23.140692632779269006);
5321: 'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5322: 'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5323: 'ESBln2','Extended').setExtended( 0.69314718055994530942);
5324: 'ESBln10','Extended').setExtended( 2.30258509299404568402);
5325: 'ESBlnP1','Extended').setExtended( 1.14472988584940017414);
5326: 'ESBlog10Base2','Extended').setExtended( 3.3219280948873623478);
5327: 'ESBlog2Base10','Extended').setExtended( 0.30102999566398119521);
5328: 'ESBlog3Base10','Extended').setExtended( 0.47712125471966243730);
5329: 'ESBlogPiBase10','Extended').setExtended( 0.4971498726941339);
5330: 'ESBlogEBase10','Extended').setExtended( 0.43429448190325182765);
5331: 'ESBPi','Extended').setExtended( 3.1415926535897932385);
5332: 'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5333: 'ESBTwopi','Extended').setExtended( 6.2831853071795864769);
5334: 'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5335: 'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5336: 'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5337: 'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5338: 'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5339: 'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5340: 'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5341: 'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5342: 'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5343: 'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5344: 'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5345: 'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5346: 'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5347: 'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5348: 'ESBlnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5349: //LongWord', 'Cardinal
5350: TBitList', 'Word
5351: Function UMul( const Num1, Num2 : LongWord) : LongWord
5352: Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5353: Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5354: Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5355: Function SameFloat( const X1, X2 : Extended) : Boolean
5356: Function FloatIsZero( const X : Extended) : Boolean
5357: Function FloatIsPositive( const X : Extended) : Boolean
5358: Function FloatIsNegative( const X : Extended) : Boolean
5359: Procedure IncLim( var B : Byte; const Limit : Byte)
5360: Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5361: Procedure IncLimW( var B : Word; const Limit : Word)
5362: Procedure IncLimI( var B : Integer; const Limit : Integer)
5363: Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5364: Procedure DecLim( var B : Byte; const Limit : Byte)
5365: Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5366: Procedure DecLimW( var B : Word; const Limit : Word)
5367: Procedure DecLimI( var B : Integer; const Limit : Integer)
5368: Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5369: Function MaxB( const B1, B2 : Byte) : Byte
5370: Function MinB( const B1, B2 : Byte) : Byte
5371: Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5372: Function MinSI( const B1, B2 : ShortInt) : ShortInt
5373: Function MaxW( const B1, B2 : Word) : Word
5374: Function MinW( const B1, B2 : Word) : Word
5375: Function esbMaxI( const B1, B2 : Integer) : Integer
5376: Function esbMinI( const B1, B2 : Integer) : Integer
5377: Function MaxL( const B1, B2 : LongInt) : LongInt
5378: Function MinL( const B1, B2 : LongInt) : LongInt
5379: Procedure SwapB( var B1, B2 : Byte)
5380: Procedure SwapSI( var B1, B2 : ShortInt)
5381: Procedure SwapW( var B1, B2 : Word)
5382: Procedure SwapI( var B1, B2 : SmallInt)
5383: Procedure SwapL( var B1, B2 : LongInt)
5384: Procedure SwapI32( var B1, B2 : Integer)
5385: Procedure SwapC( var B1, B2 : LongWord)
5386: Procedure SwapInt64( var X, Y : Int64)
5387: Function esbSign( const B : LongInt) : ShortInt
5388: Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5389: Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5390: Function Max3Word( const X1, X2, X3 : Word) : Word
5391: Function Min3Word( const X1, X2, X3 : Word) : Word
5392: Function MaxBArray( const B : array of Byte) : Byte
5393: Function MaxWArray( const B : array of Word) : Word
5394: Function MaxSIArry( const B : array of ShortInt) : ShortInt
5395: Function MaxIArray( const B : array of Integer) : Integer
5396: Function MaxLArray( const B : array of LongInt) : LongInt
5397: Function MinBArray( const B : array of Byte) : Byte
5398: Function MinWArray( const B : array of Word) : Word
5399: Function MinSIArry( const B : array of ShortInt) : ShortInt
5400: Function MinIArray( const B : array of Integer) : Integer
5401: Function MinLArray( const B : array of Longint) : Longint
5402: Function SumBArray( const B : array of Byte) : Byte
5403: Function SumBArray2( const B : array of Byte) : Word
5404: Function SumSIArry( const B : array of ShortInt) : ShortInt
5405: Function SumSIArry2( const B : array of ShortInt) : Integer
5406: Function SumWArray( const B : array of Word) : Word

```

```

5407: Function SumWArray2( const B : array of Word) : LongInt
5408: Function SumIArray( const B : array of Integer) : Integer
5409: Function SumLArray( const B : array of LongInt) : LongInt
5410: Function SumLWArray( const B : array of LongWord) : LongWord
5411: Function ESBDigits( const X : LongWord) : Byte
5412: Function BitsHighest( const X : LongWord) : Integer
5413: Function ESBBitsNeeded( const X : LongWord) : Integer
5414: Function esbGCD( const X, Y : LongWord) : LongWord
5415: Function esbLCM( const X, Y : LongInt) : Int64
5416: //Function esbLCM( const X, Y : LongInt) : LongInt
5417: Function RelativePrime( const X, Y : LongWord) : Boolean
5418: Function Get87ControlWord : TBitList
5419: Procedure Set87ControlWord( const CWord : TBitList)
5420: Procedure SwapExt( var X, Y : Extended)
5421: Procedure SwapDbl( var X, Y : Double)
5422: Procedure SwapSing( var X, Y : Single)
5423: Function esbSgn( const X : Extended) : ShortInt
5424: Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5425: Function ExtMod( const X, Y : Extended) : Extended
5426: Function ExtRem( const X, Y : Extended) : Extended
5427: Function CompMOD( const X, Y : Comp) : Comp
5428: Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5429: Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5430: Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5431: Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5432: Function MaxExt( const X, Y : Extended) : Extended
5433: Function MinExt( const X, Y : Extended) : Extended
5434: Function MaxEArray( const B : array of Extended) : Extended
5435: Function MinEArray( const B : array of Extended) : Extended
5436: Function MaxSArray( const B : array of Single) : Single
5437: Function MinSArray( const B : array of Single) : Single
5438: Function MaxCArray( const B : array of Comp) : Comp
5439: Function MinCArray( const B : array of Comp) : Comp
5440: Function SumSArray( const B : array of Single) : Single
5441: Function SumBArray( const B : array of Extended) : Extended
5442: Function SumSqEArray( const B : array of Extended) : Extended
5443: Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5444: Function SumXYEArray( const X, Y : array of Extended) : Extended
5445: Function SumCArray( const B : array of Comp) : Comp
5446: Function FactorialX( A : LongWord) : Extended
5447: Function PermutationX( N, R : LongWord) : Extended
5448: Function esbBinomialCoeff( N, R : LongWord) : Extended
5449: Function IsPositiveEArray( const X : array of Extended) : Boolean
5450: Function esbGeometricMean( const X : array of Extended) : Extended
5451: Function esbHarmonicMean( const X : array of Extended) : Extended
5452: Function ESBMean( const X : array of Extended) : Extended
5453: Function esbSampleVariance( const X : array of Extended) : Extended
5454: Function esbPopulationVariance( const X : array of Extended) : Extended
5455: Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5456: Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5457: Function GetMedian( const SortedX : array of Extended) : Extended
5458: Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5459: Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5460: Function ESBMagnitude( const X : Extended) : Integer
5461: Function ESBTan( Angle : Extended) : Extended
5462: Function ESBCot( Angle : Extended) : Extended
5463: Function ESB cosec( const Angle : Extended) : Extended
5464: Function ESB Sec( const Angle : Extended) : Extended
5465: Function ESB ArcTan( X, Y : Extended) : Extended
5466: Procedure ESB SinCos( Angle : Extended; var SinX, CosX : Extended)
5467: Function ESB ArcCos( const X : Extended) : Extended
5468: Function ESB ArcSin( const X : Extended) : Extended
5469: Function ESB ArcSec( const X : Extended) : Extended
5470: Function ESB ArcCosec( const X : Extended) : Extended
5471: Function ESB Log10( const X : Extended) : Extended
5472: Function ESB Log2( const X : Extended) : Extended
5473: Function ESB LogBase( const X, Base : Extended) : Extended
5474: Function Pow2( const X : Extended) : Extended
5475: Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5476: Function ESB InPower( const X : Extended; const N : LongInt) : Extended
5477: Function XtoY( const X, Y : Extended) : Extended
5478: Function esbTenToY( const Y : Extended) : Extended
5479: Function esbTwoToY( const Y : Extended) : Extended
5480: Function LogXtoBaseY( const X, Y : Extended) : Extended
5481: Function esbISqrt( const I : LongWord) : Longword
5482: Function ILog2( const I : LongWord) : LongWord
5483: Function IGreatestPowerOf2( const N : LongWord) : LongWord
5484: Function ESB ArCosh( X : Extended) : Extended
5485: Function ESB ArSinh( X : Extended) : Extended
5486: Function ESB ArTanh( X : Extended) : Extended
5487: Function ESB Cosh( X : Extended) : Extended
5488: Function ESB Sinh( X : Extended) : Extended
5489: Function ESB Tanh( X : Extended) : Extended
5490: Function InverseGamma( const X : Extended) : Extended
5491: Function esbGamma( const X : Extended) : Extended
5492: Function esbLnGamma( const X : Extended) : Extended
5493: Function esbBeta( const X, Y : Extended) : Extended
5494: Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5495: end;

```

```

5496:
5497: ********* Integer Huge Cardinal Utils*****
5498: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5499: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5500: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5501: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5502: Function BitCount_8( Value : byte) : integer
5503: Function BitCount_16( Value : uint16) : integer
5504: Function BitCount_32( Value : uint32) : integer
5505: Function BitCount_64( Value : uint64) : integer
5506: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5507: Procedure ( CountPrimalityTests : integer)
5508: Function gcd( a, b : THugeCardinal) : THugeCardinal
5509: Function lcm( a, b : THugeCardinal) : THugeCardinal
5510: Function isCoPrime( a, b : THugeCardinal) : boolean
5511: Function isProbablyPrime(p: THugeCardinal; OnProgress: TProgress; var wasAborted: boolean): boolean
5512: Function hasSmallFactor( p : THugeCardinal) : boolean
5513: //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool; var Prime: THugeCardinal; var
NumbersTested: integer) : boolean
5514: Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5515: Const('StandardExponent','LongInt'( 65537);
5516: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
Numbers
5517: Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5518:
5519: procedure SIRegister_xrtl_math_Integer(CL: TPPSPascalCompiler);
5520: begin
5521:   AddTypeS('TXRTLInteger', 'array of Integer
5522:   AddClassN(FindClass('TOBJECT'), 'EXRTLMathException
5523:   (FindClass('TOBJECT'), 'EXRTLExtendInvalidArgument
5524:   AddClassN(FindClass('TOBJECT'), 'EXRTLDivisionByZero
5525:   AddClassN(FindClass('TOBJECT'), 'EXRTLExpInvalidArgument
5526:   AddClassN(FindClass('TOBJECT'), 'EXRTLInvalidRadix
5527:   AddClassN(FindClass('TOBJECT'), 'EXRTLInvalidRadixDigit
5528:   AddClassN(FindClass('TOBJECT'), 'EXRTLRootInvalidArgument
5529:   'BitsPerByte','LongInt'( 8);
5530:   BitsPerDigit','LongInt'( 32);
5531:   SignBitMask ', LongWord($80000000);
5532:   Function XRTLAdjustBits( const ABits : Integer) : Integer
5533:   Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5534:   Function XRTLDATAbits( const AInteger : TXRTLInteger) : Integer
5535:   Procedure XRTLBITPosition( const BitIndex : Integer; var Index, Mask : Integer)
5536:   Procedure XRTLBITSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5537:   Procedure XRTLBITReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5538:   Function XRTLBITGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5539:   Function XRTLBITGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5540:   Function XRTLEXTEND(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5541:   Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5542:   Function XRTLSignExtend(const AInteger:TXRTLInteger;ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5543:   Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5544:   Procedure XRTLNNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5545:   Procedure XRTLOR( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5546:   Procedure XRTLAND( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5547:   Procedure XRTLXOR( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5548:   Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5549:   Procedure XRTLZero( var AInteger : TXRTLInteger)
5550:   Procedure XRTLOne( var AInteger : TXRTLInteger)
5551:   Procedure XRTLMOne( var AInteger : TXRTLInteger)
5552:   Procedure XRTLTTwo( var AInteger : TXRTLInteger)
5553:   Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5554:   Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5555:   Procedure XRTLFULLsum( const A, B, C : Integer; var Sum, Carry : Integer)
5556:   Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5557:   Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5558:   Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5559:   Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5560:   Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5561:   Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5562:   Function XRTLUMUL( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5563:   Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5564:   Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5565:   Procedure XRTLDIVMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult, RResult : TXRTLInteger)
5566:   Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5567:   Procedure XRTLSqrT( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5568:   Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5569:   Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHHighApproxResult:TXRTLInteger)
5570:   Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHHighApproxResult:TXRTLInteger);
5571:   Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5572:   Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult : TXRTLInteger)
5573:   Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5574:   Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5575:   Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5576:   Procedure XRTLSLDL(const AInteger: TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5577:   Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5578:   Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
```

```

5579: Procedure XRTLSLBR( const AInteger : TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5580: Procedure XRTLSABR( const AInteger : TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5581: Procedure XRTLRCBR( const AInteger : TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5582: Procedure XRTLSLDR( const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5583: Procedure XRTLSADR( const AInteger : TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5584: Procedure XRTLRCDR( const AInteger : TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5585: Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5586: Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5587: Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5588: Procedure XRTLFrmHex( const Value : string; var AResult : TXRTLInteger)
5589: Procedure XRTLFrmBin( const Value : string; var AResult : TXRTLInteger)
5590: Procedure XRTLFrmString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5591: Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5592: Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5593: Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5594: Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5595: Procedure XRTLSplit( const AInteger : TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5596: Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5597: Procedure XRTLMInMax( const AInteger1, AInteger2 : TXRTLInteger; var AMinResult,AMaxResult: TXRTLInteger)
5598: Procedure XRTLMIn( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5599: Procedure XRTLMIn1( const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5600: Procedure XRTLMMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5601: Procedure XRTLMMax1( const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5602: Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5603: Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5604: Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5605: Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5606: end;
5607:
5608: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5609: begin
5610:   Function JvXPMETHODSEqual( const Method1, Method2 : TMethod) : Boolean
5611:   Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5612:   Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5613:   Procedure JvXPADJUSTBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5614:   Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLns;const
      AColor:TColor;Rect:TRect)
5615:   Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5616:   Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5617:   Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5618:   Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5619:   Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5620:   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)
5621: end;
5622:
5623:
5624: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5625: begin
5626:   Function StrDec( S : String) : String
5627:   Function uIsNumeric( var S : String; var X : Float) : Boolean
5628:   Function ReadNumFromEdit( Edit : TEdit) : Float
5629:   Procedure WriteNumToFile( var F : Text; X : Float)
5630: end;
5631:
5632: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5633: begin
5634:   Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5635:   Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5636:   Procedure TeX_LeaveGraphics( Footer : Boolean)
5637:   Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5638:   Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5639:   Procedure TeX_SetGraphTitle( Title : String)
5640:   Procedure TeX_SetOxTitle( Title : String)
5641:   Procedure TeX_SetOyTitle( Title : String)
5642:   Procedure TeX_PlotOxAxis
5643:   Procedure TeX_PlotOyAxis
5644:   Procedure TeX_PlotGrid( Grid : TGrid)
5645:   Procedure TeX_WriteGraphTitle
5646:   Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5647:   Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5648:   Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5649:   Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5650:   Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5651:   Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5652:   Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5653:   Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5654:   Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5655:   Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5656:   Function Xcm( X : Float) : Float
5657:   Function Ycm( Y : Float) : Float
5658: end;
5659:
5660: *-----*)

```

```

5661: procedure SIRegister_VarRecUtils(CL: TPSPPascalCompiler);
5662: begin
5663:   TConstArray', 'array of TVarRec
5664:   Function CopyVarRec( const Item : TVarRec ) : TVarRec
5665:   Function CreateConstArray( const Elements : array of const ) : TConstArray
5666:   Procedure FinalizeVarRec( var Item : TVarRec )
5667:   Procedure FinalizeConstArray( var Arr : TConstArray )
5668: end;
5669:
5670: procedure SIRegister_StStrS(CL: TPSPPascalCompiler);
5671: begin
5672:   Function HexBS( B : Byte ) : ShortString
5673:   Function HexWS( W : Word ) : ShortString
5674:   Function HexLS( L : LongInt ) : ShortString
5675:   Function HexPtrs( P : Pointer ) : ShortString
5676:   Function BinaryBS( B : Byte ) : ShortString
5677:   Function BinaryWS( W : Word ) : ShortString
5678:   Function BinaryLS( L : LongInt ) : ShortString
5679:   Function OctalBS( B : Byte ) : ShortString
5680:   Function OctalWS( W : Word ) : ShortString
5681:   Function OctalLS( L : LongInt ) : ShortString
5682:   Function Str2Int16S( const S : ShortString; var I : SmallInt ) : Boolean
5683:   Function Str2WordS( const S : ShortString; var I : Word ) : Boolean
5684:   Function Str2LongS( const S : ShortString; var I : LongInt ) : Boolean
5685:   Function Str2RealS( const S : ShortString; var R : Double ) : Boolean
5686:   Function Str2RealS( const S : ShortString; var R : Real ) : Boolean
5687:   Function Str2ExtS( const S : ShortString; var R : Extended ) : Boolean
5688:   Function Long2StrS( L : LongInt ) : ShortString
5689:   Function Real2StrS( R : Double; Width : Byte; Places : ShortInt ) : ShortString
5690:   Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt ) : ShortString
5691:   Function ValPrepS( const S : ShortString ) : ShortString
5692:   Function CharStrS( C : AnsiChar; Len : Cardinal ) : ShortString
5693:   Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5694:   Function PadS( const S : ShortString; Len : Cardinal ) : ShortString
5695:   Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5696:   Function LeftPadS( const S : ShortString; Len : Cardinal ) : ShortString
5697:   Function TrimLeadS( const S : ShortString ) : ShortString
5698:   Function TrimTrailsS( const S : ShortString ) : ShortString
5699:   Function TrimS( const S : ShortString ) : ShortString
5700:   Function TrimSpacesS( const S : ShortString ) : ShortString
5701:   Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5702:   Function Centers( const S : ShortString; Len : Cardinal ) : ShortString
5703:   Function EntabS( const S : ShortString; TabSize : Byte ) : ShortString
5704:   Function DetabS( const S : ShortString; TabSize : Byte ) : ShortString
5705:   Function ScrambleS( const S, Key : ShortString ) : ShortString
5706:   Function SubstituteS( const S, FromStr,ToStr : ShortString ) : ShortString
5707:   Function Filters( const S, Filters : ShortString ) : ShortString
5708:   Function CharExistsS( const S : ShortString; C : AnsiChar ) : Boolean
5709:   Function CharCountS( const S : ShortString; C : AnsiChar ) : Byte
5710:   Function WordCounts( const S, WordDelims : ShortString ) : Cardinal
5711:   Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal ) : Boolean
5712:   Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString ) : ShortString
5713:   Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar ) : Cardinal
5714:   Function AsciiPositionS( N : Cardinal; const S, WordDelims : ShortString; Quote : AnsiChar; var Pos : Cardinal ) : Boolean
5715:   Function ExtractAsciiS( N : Cardinal; const S, WordDelims : ShortString; Quote : AnsiChar ) : ShortString
5716:   Procedure WordWrapsS( const Inst : ShortString; var OutSt,Overlap : ShortString );
Margin:Cardinal;PadToMargin:Boolean)
5717:   Function CompStringS( const S1, S2 : ShortString ) : Integer
5718:   Function CompUCStringS( const S1, S2 : ShortString ) : Integer
5719:   Function SoundexS( const S : ShortString ) : ShortString
5720:   Function MakeLetterSetS( const S : ShortString ) : Longint
5721:   Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable )
5722:   Function BMSearchS( var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;
5723:   Function BMSearchUCS( var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
Pos:Cardinal):Bool;
5724:   Function DefaultExtensionS( const Name, Ext : ShortString ) : ShortString
5725:   Function ForceExtensionS( const Name, Ext : ShortString ) : ShortString
5726:   Function JustFilenameS( const PathName : ShortString ) : ShortString
5727:   Function JustNameS( const PathName : ShortString ) : ShortString
5728:   Function JustExtensionS( const Name : ShortString ) : ShortString
5729:   Function JustPathnameS( const PathName : ShortString ) : ShortString
5730:   Function AddBackSlashS( const DirName : ShortString ) : ShortString
5731:   Function CleanPathNameS( const PathName : ShortString ) : ShortString
5732:   Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal ) : Boolean
5733:   Function CommaizeS( L : LongInt ) : ShortString
5734:   Function CommaizeChS( L : LongInt; Ch : AnsiChar ) : ShortString
5735:   Function FloatFormS( const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
DecPt:Char):ShortString;
5736:   Function LongIntFormS( const Mask:ShortString;L:LongInt;const LtCurr,
RtCurr:ShortString;Sep:AnsiChar):ShortString;
5737:   Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal ) : Boolean
5738:   Function StrStPosS( const P, S : ShortString; var Pos : Cardinal ) : Boolean
5739:   Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal ) : ShortString
5740:   Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal ) : ShortString
5741:   Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal ) : ShortString
5742:   Function StrChDeleteS( const S : ShortString; Pos : Cardinal ) : ShortString
5743:   Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal ) : ShortString
5744:   Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal ) : Boolean

```

```

5745: Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal ) : Boolean
5746: Function CopyLeftS( const S : ShortString; Len : Cardinal ) : ShortString
5747: Function CopyMidS( const S : ShortString; First, Len : Cardinal ) : ShortString
5748: Function CopyRights( const S : ShortString; First : Cardinal ) : shortString
5749: Function CopyRightAbsS( const S : shortString; NumChars : Cardinal ) : ShortString
5750: Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
SubString:ShortString):Bool;
5751: Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
SubStr:ShortString):Bool;
5752: Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5753: Function DeleteFromToWords(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5754: Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean ) : ShortString
5755: Function DeleteWithinS( const S, Delimiter : ShortString ) : ShortString
5756: Function ExtractTokensS(const S,
Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5757: Function IsChAlphas( C : Char ) : Boolean
5758: Function IsChNumericS( C : Char; const Numbers : ShortString ) : Boolean
5759: Function IsChAlphaNumerics( C : Char; const Numbers : shortString ) : Boolean
5760: Function IsStrAlphas( const S : Shortstring ) : Boolean
5761: Function IsStrNumericS( const S, Numbers : ShortString ) : Boolean
5762: Function IsStrAlphaNumericS( const S, Numbers : ShortString ) : Boolean
5763: Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal ) : Boolean
5764: Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal ) : Boolean
5765: Function LastStringS( const S, AString : ShortString; var Position : Cardinal ) : Boolean
5766: Function LeftTrimCharsS( const S, Chars : ShortString ) : ShortString
5767: Function KeepCharsS( const S, Chars : ShortString ) : ShortString
5768: Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen : Cardinal):ShortString;
5769: Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
Replacements:Cardinal):ShortString;
5770: Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5771: Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
Replacements:Cardinal):ShortString
5772: Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
Replacements:Cardinal):ShortString
5773: Function RightTrimCharsS( const S, Chars : ShortString ) : ShortString
5774: Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5775: Function TrimCharsS( const S, Chars : ShortString ) : ShortString
5776: Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5777: end;
5778:
5779:
5780: *****unit uPSI_StUtils; from SysTools4***** ****
5781: Function SignL( L : LongInt ) : Integer
5782: Function SignF( F : Extended ) : Integer
5783: Function MinWord( A, B : Word ) : Word
5784: Function MidWord( W1, W2, W3 : Word ) : Word
5785: Function MaxWord( A, B : Word ) : Word
5786: Function MinLong( A, B : LongInt ) : LongInt
5787: Function MidLong( L1, L2, L3 : LongInt ) : LongInt
5788: Function MaxLong( A, B : LongInt ) : LongInt
5789: Function MinFloat( F1, F2 : Extended ) : Extended
5790: Function MidFloat( F1, F2, F3 : Extended ) : Extended
5791: Function MaxFloat( F1, F2 : Extended ) : Extended
5792: Function MakeInteger16( H, L : Byte ) : SmallInt
5793: Function MakeWordS( H, L : Byte ) : Word
5794: Function SwapNibble( B : Byte ) : Byte
5795: Function SwapWord( L : LongInt ) : LongInt
5796: Procedure SetFlag( var Flags : Word; FlagMask : Word )
5797: Procedure ClearFlag( var Flags : Word; FlagMask : Word )
5798: Function FlagIsSet( Flags, FlagMask : Word ) : Boolean
5799: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte )
5800: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte )
5801: Function ByteFlagIsSet( Flags, FlagMask : Byte ) : Boolean
5802: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt )
5803: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt )
5804: Function LongFlagIsSet( Flags, FlagMask : LongInt ) : Boolean
5805: Procedure ExchangeBytes( var I, J : Byte )
5806: Procedure ExchangeWords( var I, J : Word )
5807: Procedure ExchangeLongInts( var I, J : LongInt )
5808: Procedure ExchangeStructs( var I, J, Size : Cardinal )
5809: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word )
5810: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal )
5811: Function AddWordToPtr( P : __Pointer; W : Word ) : __Pointer
5812: //*****uPSI_StFIN;*****
5813: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5814: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5815: Function BondDuration( Settlement,Maturity:TStDate;Rate,
Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5816: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
Extended
5817: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5818: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5819: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis ) : LongInt

```

```

5820: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer ) : Extended
5821: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis) : Extended;
5822: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer ) : Extended
5823: Function DollarToDecimalText( DecDollar : Extended ) : string
5824: Function DollarToFraction( DecDollar : Extended; Fraction : Integer ) : Extended
5825: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer ) : string
5826: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency ) : Extended
5827: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
    PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5828: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double ) : Extended
5829: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer ) : Extended
5830: Function InterestRateS(NPeriods:Int;Pmt,PV,
    FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5831: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended ) : Extended
5832: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended ) : Extended
5833: Function IsCardValid( const S : string ) : Boolean
5834: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
    Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5835: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5836: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5837: Function NetPresentValueS( Rate : Extended; const Values : array of Double ) : Extended
5838: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer ) : Extended
5839: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency ) : Extended
5840: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5841: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5842: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
    : TStPaymentTime): Extended
5843: Function Periods( Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5844: Function PresentValueS( Rate : Extended; NPeriods : Integer; Pmt, FV : Extended; Frequency : TStFrequency;
    Timing : TStPaymentTime): Extended
5845: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5846: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean ) : Extended
5847: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5848: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5849: Function TBillyield( Settlement, Maturity : TStDate; Price : Extended ) : Extended
5850: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
    Factor : Extended; NoSwitch : boolean ) : Extended
5851: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5852: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
    Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5853: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5854:
5855: //*****unit uPSI_StAstroP;
5856: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray )
5857: //*****unit unit uPSI_StStat; Statistic Package of SysTools*****
5858: Function AveDev( const Data : array of Double ) : Double
5859: Function AveDev16( const Data, NData : Integer ) : Double
5860: Function Confidence( Alpha, StandardDev : Double; Size : LongInt ) : Double
5861: Function Correlation( const Data1, Data2 : array of Double ) : Double
5862: Function Correlation16( const Data1, Data2, NData : Integer ) : Double
5863: Function Covariance( const Data1, Data2 : array of Double ) : Double
5864: Function Covariance16( const Data1, Data2, NData : Integer ) : Double
5865: Function DevSq( const Data : array of Double ) : Double
5866: Function DevSq16( const Data, NData : Integer ) : Double
5867: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5868: //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5869: Function GeometricMeanS( const Data : array of Double ) : Double
5870: Function GeometricMean16( const Data, NData : Integer ) : Double
5871: Function HarmonicMeanS( const Data : array of Double ) : Double
5872: Function HarmonicMean16( const Data, NData : Integer ) : Double
5873: Function Largest( const Data : array of Double; K : Integer ) : Double
5874: Function Largest16( const Data, NData : Integer; K : Integer ) : Double
5875: Function MedianS( const Data : array of Double ) : Double
5876: Function Median16( const Data, NData : Integer ) : Double
5877: Function Mode( const Data : array of Double ) : Double
5878: Function Mode16( const Data, NData : Integer ) : Double
5879: Function Percentile( const Data : array of Double; K : Double ) : Double
5880: Function Percentile16( const Data, NData : Integer; K : Double ) : Double
5881: Function PercentRank( const Data : array of Double; X : Double ) : Double
5882: Function PercentRank16( const Data, NData : Integer; X : Double ) : Double
5883: Function Permutations( Number, NumberChosen : Integer ) : Extended
5884: Function Combinations( Number, NumberChosen : Integer ) : Extended
5885: Function Factorials( N : Integer ) : Extended
5886: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean ) : Integer
5887: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean ) : Integer
5888: Function Smallest( const Data : array of Double; K : Integer ) : Double
5889: Function Smallest16( const Data, NData : Integer; K : Integer ) : Double
5890: Function TrimMean( const Data : array of Double; Percent : Double ) : Double
5891: Function TrimMean16( const Data, NData : Integer; Percent : Double ) : Double
5892: AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5893:     + '1 : Double; R2 : Double; sigma :Double; SSr: double; SSE: Double; F0 : Double; df : Integer;end
5894: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
    LF:TStLinEst;ErrorStats:Bool;
5895: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
    LF:TStLinEst;ErrorStats:Bool;
5896: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5897: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5898: Function Intercept( const KnownY : array of Double; const KnownX : array of Double ) : Double
5899: Function RSquared( const KnownY : array of Double; const KnownX : array of Double ) : Double

```

```

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

```

```

5989: Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean ) : TStTime
5990: Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt)
5991: Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5992: Function DateToString( const Month : Integer ) : string
5993: Function DateStringToTStDate( DT : TDateTime ) : TStDate
5994: Function StDateToDateTime( D : TStDate ) : TDateTime
5995: Function StTimeToDateTime( T : TStTime ) : TDateTime
5996: Function Convert2ByteDate( TwoByteDate : Word ) : TStDate
5997: Function Convert4ByteDate( FourByteDate : TStDate ) : Word
5998:
5999: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
6000: Function DateStringHMSToAstJD( const Picture, DS : string; H, M, S, Epoch : integer ) : Double
6001: Function MonthToString( const Month : Integer ) : string
6002: Function DateStringToTStDate( const Picture, S : string; Epoch : Integer ) : TStDate
6003: Function DateStringToDMY( const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
6004: Function StDateToString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
6005: Function DayOfWeekToString( const WeekDay : TStDayType) : string
6006: Function DMYToDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string;
6007: Function CurrentDateString( const Picture : string; Pack : Boolean) : string
6008: Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
6009: Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer ) : Boolean
6010: Function TimeStringToTStTime( const Picture, S : string ) : TStTime
6011: Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean ) : string
6012: Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean ) : string
6013: Function DateStringIsBlank( const Picture, S : string ) : Boolean
6014: Function InternationalDate( ForceCentury : Boolean ) : string
6015: Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean ) : string
6016: Function InternationalTime( ShowSeconds : Boolean ) : string
6017: Procedure ResetInternationalInfo
6018:
6019: procedure SIRegister_StBase(CL: TPSPascalCompiler);
6020: Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer ) : Boolean
6021: Function AnsiUpperCaseShort32( const S : string ) : string
6022: Function AnsiCompareTextShort32( const S1, S2 : string ) : Integer
6023: Function AnsiCompareStrShort32( const S1, S2 : string ) : Integer
6024: Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer ) : Longint
6025: Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt) : Longint
6026: Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte )
6027: Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal )
6028: Function Upcase( C : AnsiChar ) : AnsiChar
6029: Function LoCase( C : AnsiChar ) : AnsiChar
6030: Function CompareLetterSets( Set1, Set2 : LongInt ) : Cardinal
6031: Function CompStruct( const S1, S2, Size : Cardinal ) : Integer
6032: Function Search( const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
6033: Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6034: Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6035: Function IsOrInheritsFrom( Root, Candidate : TClass ) : boolean
6036: Procedure RaiseContainerError( Code : longint )
6037: Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const )
6038: Function ProductOverflow( A, B : LongInt ) : Boolean
6039: Function StNewStr( S : string ) : PShortString
6040: Procedure StDisposeStr( PS : PShortString )
6041: Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer )
6042: Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer )
6043: Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer )
6044: Procedure RaiseSError( ExceptionClass : ESTExceptionClass; Code : LongInt )
6045: Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt )
6046: Procedure RaiseStWin32ErrorEx( ExceptionClass : ESTExceptionClass; Code : LongInt; Info : string )
6047:
6048: procedure SIRegister_usvd(CL: TPSPascalCompiler);
6049: begin
6050: Procedure SV_DecomP( A : TMatrix; Lb, Ubl, Ub2 : Integer; S : TVector; V : TMatrix )
6051: Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float )
6052: Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ubl,Ub2:Integer;X:TVector);
6053: Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ubl, Ub2 : Integer; A : TMatrix )
6054: Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6055: end;
6056:
6057: //*****unit unit ; StMath Package of SysTools*****
6058: Function IntPowerS( Base : Extended; Exponent : Integer ) : Extended
6059: Function PowerS( Base, Exponent : Extended ) : Extended
6060: Function StInvCos( X : Double ) : Double
6061: Function StInvSin( Y : Double ) : Double
6062: Function StInvTan2( X, Y : Double ) : Double
6063: Function StTan( A : Double ) : Double
6064: Procedure DumpException; //unit StExpEng;
6065: Function HexifyBlock( var Buffer, BufferSize : Integer ) : string
6066:
6067: //*****unit unit ; StCRC Package of SysTools*****
6068: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
6069: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
6070: Function Adler32OfFile( FileName : AnsiString ) : LongInt
6071: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
6072: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
6073: Function Crc16OfFile( FileName : AnsiString ) : Cardinal
6074: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
6075: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
6076: Function Crc32OfFile( FileName : AnsiString ) : LongInt
6077: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal

```

```

6078: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
6079: Function InternetSumOfFile( FileName : AnsiString ) : Cardinal
6080: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
6081: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
6082: Function Kermit16OfFile( FileName : AnsiString ) : Cardinal
6083:
6084: //*****unit unit ; StBCD Package of SysTools*****
6085: Function AddBcd( const B1, B2 : TbcdS ) : TbcdS
6086: Function SubBcd( const B1, B2 : TbcdS ) : TbcdS
6087: Function MulBcd( const B1, B2 : TbcdS ) : TbcdS
6088: Function DivBcd( const B1, B2 : TbcdS ) : TbcdS
6089: Function ModBcd( const B1, B2 : TbcdS ) : TbcdS
6090: Function NegBcd( const B : TbcdS ) : TbcdS
6091: Function AbsBcd( const B : TbcdS ) : TbcdS
6092: Function FracBcd( const B : TbcdS ) : TbcdS
6093: Function IntBcd( const B : TbcdS ) : TbcdS
6094: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal ) : TbcdS
6095: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal ) : TbcdS
6096: Function ValBcd( const S : string ) : TbcdS
6097: Function LongBcd( L : LongInt ) : TbcdS
6098: Function ExtBcd( E : Extended ) : TbcdS
6099: Function ExpBcd( const B : TbcdS ) : TbcdS
6100: Function LnBcd( const B : TbcdS ) : TbcdS
6101: Function IntPowBcd( const B : TbcdS; E : LongInt ) : TbcdS
6102: Function PowBcd( const B, E : TbcdS ) : TbcdS
6103: Function SqrtBcd( const B : TbcdS ) : TbcdS
6104: Function CmpBcd( const B1, B2 : TbcdS ) : Integer
6105: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal ) : Boolean
6106: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal ) : Boolean
6107: Function IsIntBcd( const B : TbcdS ) : Boolean
6108: Function TruncBcd( const B : TbcdS ) : LongInt
6109: Function BcdExt( const B : TbcdS ) : Extended
6110: Function RoundBcd( const B : TbcdS ) : LongInt
6111: Function StrBcd( const B : TbcdS; Width, Places : Cardinal ) : string
6112: Function StrExpBcd( const B : TbcdS; Width : Cardinal ) : string
6113: Function FormatBcd( const Format : string; const B : TbcdS ) : string
6114: Function StrGeneralBcd( const B : TbcdS ) : string
6115: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6116: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte )
6117:
6118: //*****unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools*****
6119: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings )
6120: Function StFieldTypeToStr( FieldType : TStSchemaFieldType ) : AnsiString
6121: Function StStrToFieldType( const S : AnsiString ) : TStSchemaFieldType
6122: Function StDeEscape( const EscStr : AnsiString ) : Char
6123: Function StDoEscape( Delim : Char ) : AnsiString
6124: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char ) : AnsiString
6125: Function AnsiHashText( const S : string; Size : Integer ) : Integer
6126: Function AnsiHashStr( const S : string; Size : Integer ) : Integer
6127: Function AnsiELFHashText( const S : string; Size : Integer ) : Integer
6128: Function AnsiELFHashStr( const S : string; Size : Integer ) : Integer
6129:
6130: //*****unit unit ; StNetCon Package of SysTools*****
6131: with AddClassN(FindClass('TStComponent'), 'TStNetConnection') do begin
6132:   Constructor Create( AOwner : TComponent )
6133:   Function Connect : DWord
6134:   Function Disconnect : DWord
6135:   RegisterProperty('Password', 'String', iptrw);
6136:   Property('UserName', 'String', iptrw);
6137:   Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6138:   Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6139:   Property('LocalDevice', 'String', iptrw);
6140:   Property('ServerName', 'String', iptrw);
6141:   Property('ShareName', 'String', iptrw);
6142:   Property('OnConnect', 'TNotifyEvent', iptrw);
6143:   Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6144:   Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6145:   Property('OnDisconnect', 'TNotifyEvent', iptrw);
6146:   Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6147:   Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6148: end;
6149: //*****Thread Functions Context of Win API --- more objects in SyncObjs.pas
6150: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6151: Procedure InitializeCriticalSection( var lpCriticalSection : TRTCriticalSection )
6152: Procedure EnterCriticalSection( var lpCriticalSection : TRTCriticalSection )
6153: Procedure LeaveCriticalSection( var lpCriticalSection : TRTCriticalSection )
6154: Function InitializeCriticalSectionAndSpinCount(var
lpCriticalSection:TRTCriticalSection;dwSpinCount:DWORD):BOOL;
6155: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTCriticalSection;dwSpinCount:DWORD):DWORD;
6156: Function TryEnterCriticalSection( var lpCriticalSection : TRTCriticalSection ) : BOOL
6157: Procedure DeleteCriticalSection( var lpCriticalSection : TRTCriticalSection )
6158: Function GetThreadContext( hThread : THHandle; var lpContext : TContext ) : BOOL
6159: Function SetThreadContext( hThread : THHandle; const lpContext : TContext ) : BOOL
6160: Function SuspendThread( hThread : THHandle ) : DWORD
6161: Function ResumeThread( hThread : THHandle ) : DWORD
6162: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THHandle
6163: Function GetCurrentThread : THHandle
6164: Procedure ExitThread( dwExitCode : DWORD )
6165: Function TerminateThread( hThread : THHandle; dwExitCode : DWORD ) : BOOL

```

```

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

```

```

6250: STIF_SUPPORT_HEX', 'LongWord( $00000001);
6251: Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6252: Function StrNCopy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6253: Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6254: Function PathAddBackslash( pszPath : PChar) : PChar
6255: Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6256: Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6257: Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6258: Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6259: Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6260: Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6261: Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6262: Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6263: Function PathFileExists( pszPath : PChar) : BOOL
6264: Function PathFindExtension( pszPath : PChar) : PChar
6265: Function PathFindFileName( pszPath : PChar) : PChar
6266: Function PathFindNextComponent( pszPath : PChar) : PChar
6267: Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6268: Function PathGetArgs( pszPath : PChar) : PChar
6269: Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6270: Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6271: Function PathGetCharType( ch : Char) : UINT
6272: GCT_INVALID', 'LongWord( $0000);
6273: GCT_LFNCHAR', 'LongWord( $0001);
6274: GCT_SHORTCHAR', 'LongWord( $0002);
6275: GCT_WILD', 'LongWord( $0004);
6276: GCT_SEPARATOR', 'LongWord( $0008);
6277: Function PathGetDriveNumber( pszPath : PChar) : Integer
6278: Function PathIsDirectory( pszPath : PChar) : BOOL
6279: Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6280: Function PathIsFileSpec( pszPath : PChar) : BOOL
6281: Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6282: Function PathIsRelative( pszPath : PChar) : BOOL
6283: Function PathIsRoot( pszPath : PChar) : BOOL
6284: Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6285: Function PathIsUNC( pszPath : PChar) : BOOL
6286: Function PathIsNetworkPath( pszPath : PChar) : BOOL
6287: Function PathIsUNCServer( pszPath : PChar) : BOOL
6288: Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6289: Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6290: Function PathIsURL( pszPath : PChar) : BOOL
6291: Function PathMakePretty( pszPath : PChar) : BOOL
6292: Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6293: Function PathParseIconLocation( pszIconFile : PChar) : Integer
6294: Procedure PathQuoteSpaces( lpsz : PChar)
6295: Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6296: Procedure PathRemoveArgs( pszPath : PChar)
6297: Function PathRemoveBackslash( pszPath : PChar) : PChar
6298: Procedure PathRemoveBlanks( pszPath : PChar)
6299: Procedure PathRemoveExtension( pszPath : PChar)
6300: Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6301: Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6302: Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6303: Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6304: Function PathSkipRoot( pszPath : PChar) : PChar
6305: Procedure PathStripPath( pszPath : PChar)
6306: Function PathStripToRoot( pszPath : PChar) : BOOL
6307: Procedure PathUnquoteSpaces( lpsz : PChar)
6308: Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6309: Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6310: Function PathIsSystemFolder( pszPath : PChar; dwAttrib : DWORD) : BOOL
6311: Procedure PathUndecorate( pszPath : PChar)
6312: Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6313: URL_SCHEME_INVALID', 'LongInt'( - 1);
6314: URL_SCHEME_UNKNOWN', 'LongInt'( 0);
6315: URL_SCHEME_FTP', 'LongInt'( 1);
6316: URL_SCHEME_HTTP', 'LongInt'( 2);
6317: URL_SCHEME_GOPHER', 'LongInt'( 3);
6318: URL_SCHEME_MAILTO', 'LongInt'( 4);
6319: URL_SCHEME_NEWS', 'LongInt'( 5);
6320: URL_SCHEME_NNTP', 'LongInt'( 6);
6321: URL_SCHEME_TELNET', 'LongInt'( 7);
6322: URL_SCHEME_WAIS', 'LongInt'( 8);
6323: URL_SCHEME_FILE', 'LongInt'( 9);
6324: URL_SCHEME_MK', 'LongInt'( 10);
6325: URL_SCHEME_HTTPS', 'LongInt'( 11);
6326: URL_SCHEME_SHELL', 'LongInt'( 12);
6327: URL_SCHEME_SNEWS', 'LongInt'( 13);
6328: URL_SCHEME_LOCAL', 'LongInt'( 14);
6329: URL_SCHEME_JAVASCRIPT', 'LongInt'( 15);
6330: URL_SCHEME_VBSCRIPT', 'LongInt'( 16);
6331: URL_SCHEME_ABOUT', 'LongInt'( 17);
6332: URL_SCHEME_RES', 'LongInt'( 18);
6333: URL_SCHEME_MAXVALUE', 'LongInt'( 19);
6334: URL_SCHEME', 'Integer
6335: URL_PART_NONE', 'LongInt'( 0);
6336: URL_PART_SCHEME', 'LongInt'( 1);
6337: URL_PART_HOSTNAME', 'LongInt'( 2);
6338: URL_PART_USERNAME', 'LongInt'( 3);

```

```

6339: URL_PART_PASSWORD', 'LongInt'( 4);
6340: URL_PART_PORT', 'LongInt'( 5);
6341: URL_PART_QUERY', 'LongInt'( 6);
6342: URL_PART', 'DWORD
6343: URLIS_URL', 'LongInt'( 0);
6344: URLIS_OPAQUE', 'LongInt'( 1);
6345: URLIS_NOHISTORY', 'LongInt'( 2);
6346: URLIS_FILEURL', 'LongInt'( 3);
6347: URLIS_APPLICABLE', 'LongInt'( 4);
6348: URLIS_DIRECTORY', 'LongInt'( 5);
6349: URLIS_HASQUERY', 'LongInt'( 6);
6350: TUrlIs', 'DWORD
6351: URL_UNESCAPE', 'LongWord( $10000000);
6352: URL_ESCAPE_UNSAFE', 'LongWord( $20000000);
6353: URL_PLUGGABLE_PROTOCOL', 'LongWord( $40000000);
6354: URL_WININET_COMPATIBILITY', 'LongWord( DWORD( $80000000 ) );
6355: URL_DONT_ESCAPE_EXTRA_INFO', 'LongWord( $02000000);
6356: URL_ESCAPE_SPACES_ONLY', 'LongWord( $04000000);
6357: URL_DONT_SIMPLIFY', 'LongWord( $08000000);
6358: URL_NO_META', 'longword( URL_DONT_SIMPLIFY);
6359: URL_UNESCAPE_INPLACE', 'LongWord( $00100000);
6360: URL_CONVERT_IF_DOSPATH', 'LongWord( $00200000);
6361: URL_UNESCAPE_HIGH_ANSI_ONLY', 'LongWord( $00400000);
6362: URL_INTERNAL_PATH', 'LongWord( $00800000);
6363: URL_FILE_USE_PATHURL', 'LongWord( $00010000);
6364: URL_ESCAPE_PERCENT', 'LongWord( $00001000);
6365: URL_ESCAPE_SEGMENT_ONLY', 'LongWord( $00002000);
6366: URL_PARTFLAG_KEEPSCHEME', 'LongWord( $00000001);
6367: URL_APPLY_DEFAULT', 'LongWord( $00000001);
6368: URL_APPLY_GUESSSCHEME', 'LongWord( $00000002);
6369: URL_APPLY_GUESSFILE', 'LongWord( $00000004);
6370: URL_APPLY_FORCEAPPLY', 'LongWord( $00000008);
6371: Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL ) : Integer
6372: Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD) : HRESULT;
6373: Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD) : HRESULT;
6374: Function UrlIsOpaque( pszUrl : PChar ) : BOOL
6375: Function UrlIsNoHistory( pszURL : PChar ) : BOOL
6376: Function UrlIsFileUrl( pszURL : PChar ) : BOOL
6377: Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs ) : BOOL
6378: Function UrlGetLocation( psz1 : PChar ) : PChar
6379: Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD ) : HRESULT
6380: Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6381: Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6382: Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6383: Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6384: Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6385: Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD ) : HRESULT
6386: Function HashData( pData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6387: Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD ) : HRESULT
6388: Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD ) : HRESULT
6389: Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6390: Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6391: Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar ) : DWORD
6392: Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD ) : Longint
6393: Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
pcchValueName:DWORD;pdwType:DWORD; pvData : _Pointer; pcbData : DWORD ) : Longint
6394: Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcch.MaxValueNameLen:DWORD):Longint;
6395: Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD ) : DWORD
6396: Function SHRegGetPath(hKey:HKEY; pkszSubKey, pkszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6397: Function SHRegSetPath( hKey:HKEY; pkszSubKey, pkszValue, pkszPath : PChar; dwFlags : DWORD): DWORD
6398: SHREGDEL_DEFAULT', 'LongWord( $00000000);
6399: SHREGDEL_HKCU', 'LongWord( $00000001);
6400: SHREGDEL_HKLM', 'LongWord( $00000010);
6401: SHREGDEL_BOTH', 'LongWord( $00000011);
6402: SHREGENUM_DEFAULT', 'LongWord( $00000000);
6403: SHREGENUM_HKCU', 'LongWord( $00000001);
6404: SHREGENUM_HKLM', 'LongWord( $00000010);
6405: SHREGENUM_BOTH', 'LongWord( $00000011);
6406: SHREGSET_HKCU', 'LongWord( $00000001);
6407: SHREGSET_FORCE_HKCU', 'LongWord( $00000002);
6408: SHREGSET_HKLM', 'LongWord( $00000004);
6409: SHREGSET_FORCE_HKLM', 'LongWord( $00000008);
6410: TSHRegDelFlags', 'DWORD
6411: TSHRegEnumFlags', 'DWORD
6412: HUSKEY', 'THandle
6413: ASSOCF_INIT_NOREMAPCLSID', 'LongWord( $00000001);
6414: ASSOCF_INIT_BYEXENAME', 'LongWord( $00000002);
6415: ASSOCF_OPEN_BYEXENAME', 'LongWord( $00000002);
6416: ASSOCF_INIT_DEFAULTTOSTAR', 'LongWord( $00000004);
6417: ASSOCF_INIT_DEFAULTTOFOLDER', 'LongWord( $00000008);
6418: ASSOCF_NOUSERSETTINGS', 'LongWord( $00000010);
6419: ASSOCF_NOTRUNCATE', 'LongWord( $00000020);
6420: ASSOCF_VERIFY', 'LongWord( $00000040);
6421: ASSOCF_REMAPRUNDLL', 'LongWord( $00000080);
6422: ASSOCF_NOFIXUPS', 'LongWord( $00000100);
6423: ASSOCF_IGNOREBASECLASS', 'LongWord( $00000200);
6424: ASSOCF', 'DWORD
6425: ASSOCSTR_COMMAND', 'LongInt'( 1);
6426: ASSOCSTR_EXECUTABLE', 'LongInt'( 2);

```

```

6427: ASSOCSTR_FRIENDLYDOCNAME', 'LongInt'( 3);
6428: ASSOCSTR_FRIENDLYAPPNAME', 'LongInt'( 4);
6429: ASSOCSTR_NOOPEN', 'LongInt'( 5);
6430: ASSOCSTR_SHELLNEWVALUE', 'LongInt'( 6);
6431: ASSOCSTR_DDECOMMAND', 'LongInt'( 7);
6432: ASSOCSTR_DDEIFEXEC', 'LongInt'( 8);
6433: ASSOCSTR_DDEAPPLICATION', 'LongInt'( 9);
6434: ASSOCSTR_DDETOPIC', 'LongInt'( 10);
6435: ASSOCSTR_INFOTIP', 'LongInt'( 11);
6436: ASSOCSTR_MAX', 'LongInt'( 12);
6437: ASSOCSTR', 'DWORD
6438: ASSOCKEY_SHELLEXECCLASS', 'LongInt'( 1);
6439: ASSOCKEY_APP', 'LongInt'( 2);
6440: ASSOCKEY_CLASS', 'LongInt'( 3);
6441: ASSOCKEY_BASECLASS', 'LongInt'( 4);
6442: ASSOCKEY_MAX', 'LongInt'( 5);
6443: ASSOCKEY', 'DWORD
6444: ASSOCDATA_MSIDEScriptor', 'LongInt'( 1);
6445: ASSOCDATA_NOACTIVATEHANDLER', 'LongInt'( 2);
6446: ASSOCDATA_QUERYCLASSTORE', 'LongInt'( 3);
6447: ASSOCDATA_HASPERUSERASSOC', 'LongInt'( 4);
6448: ASSOCDATA_MAX', 'LongInt'( 5);
6449: ASSOCDATA', 'DWORD
6450: ASSOCENUM_NONE', 'LongInt'( 0);
6451: ASSOCENUM', 'DWORD
6452: SID_IQueryAssociations', 'String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6453: SHACF_DEFAULT $00000000;
6454: SHACF_FILESYSTEM', 'LongWord( $00000001);
6455: SHACF_URLHISTORY', 'LongWord( $00000002);
6456: SHACF_URLMRU', 'LongWord( $00000004);
6457: SHACF_USETAB', 'LongWord( $00000008);
6458: SHACF_FILESYS_ONLY', 'LongWord( $00000010);
6459: SHACF_AUTOSUGGEST_FORCE_ON', 'LongWord( $10000000);
6460: SHACF_AUTOSUGGEST_FORCE_OFF', 'LongWord( $20000000);
6461: SHACF_AUTOAPPEND_FORCE_ON', 'LongWord( $40000000);
6462: SHACF_AUTOAPPEND_FORCE_OFF', 'LongWord( DWORD( $80000000 ) );
6463: Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD ) : HRESULT
6464: Procedure SHSetThreadRef( punk : IUnknown )
6465: Procedure SHGetThreadRef( out ppunk : IUnknown )
6466: CTF_INSIST', 'LongWord( $00000001);
6467: CTF_THREAD_REF', 'LongWord( $00000002);
6468: CTF_PROCESS_REF', 'LongWord( $00000004);
6469: CTF_COINIT', 'LongWord( $00000008);
6470: Function SHCreateShellPalette( hdc : HDC ) : HPALETTE
6471: Procedure ColorRGBtoHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD )
6472: Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD ) : TColorRef
6473: Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean ) : TColorRef
6474: Function GetSysColorBrush( nIndex : Integer ) : HBRUSH
6475: Function DrawFocusRect( hdc : HDC; const lprc : TRect ) : BOOL
6476: Function FillRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH ) : Integer
6477: Function FrameRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH ) : Integer
6478: Function InvertRect( hdc : HDC; const lprc : TRect ) : BOOL
6479: Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer ) : BOOL
6480: Function SetRectEmpty( var lprc : TRect ) : BOOL
6481: Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect ) : BOOL
6482: Function InflateRect( var lprc : TRect; dx, dy : Integer ) : BOOL
6483: Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect ) : BOOL
6484: Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect ) : BOOL
6485:
6486: Function InitializeFlatSB( hWnd : HWND ) : Bool
6487: Procedure UninitializeFlatSB( hWnd : HWND )
6488: Function FlatSB_GetScrollProp( pl : HWND; propIndex : Integer; p3 : PInteger ) : Bool
6489: Function FlatSB_SetScrollProp( pl : HWND; index : Integer; newValue : Integer; p4 : Bool ) : Bool
6490: Function GET_APPCOMMAND_LPARAM( lParam : Integer ) : Word //of JvWin32
6491: Function GET_DEVICE_LPARAM( lParam : Integer ) : Word
6492: Function GET_MOUSEORKEY_LPARAM( lParam : Integer ) : Integer
6493: Function GET_FLAGS_LPARAM( lParam : Integer ) : Word
6494: Function GET_KEYSTATE_LPARAM( lParam : Integer ) : Word
6495:
6496:
6497: // **** 204 unit uPSI_ShellAPI;
6498: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT ) : UINT
6499: Function DragQueryPoint( Drop : HDROP; var Point : TPoint ) : BOOL
6500: Procedure DragFinish( Drop : HDROP )
6501: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL )
6502: Function ShellExecute( hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer ):HINST
6503: Function FindExecutable( FileName, Directory : PChar; Result : PChar ) : HINST
6504: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON ) : Integer
6505: Function DuplicateIcon( hInst : HINST; Icon : HICON ) : HICON
6506: Function ExtractAssociatedIcon( hInst : HINST; lpiIconPath : PChar; var lpiIcon : Word ) : HICON
6507: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT ) : HICON
6508: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData ) : UINT
6509: Function DoEnvironmentSubst( szString : PChar; cbString : UINT ) : DWORD
6510: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6511: Procedure SHFreeNameMappings( hNameMappings : THandle )
6512:
6513: DLLVER_PLATFORM_WINDOWS', 'LongWord( $00000001);
6514: DLLVER_PLATFORM_NT', 'LongWord( $00000002);
6515: DLLVER_MAJOR_MASK', 'LongWord( Int64 ( $FFFF000000000000 ) );

```

```

6516: DLLVER_MINOR_MASK', 'LongWord( Int64 ( $0000FFFF00000000 ) );
6517: DLLVER_BUILD_MASK', 'LongWord( Int64 ( $00000000FFFF0000 ) );
6518: DLLVER_QFE_MASK', 'LongWord( Int64 ( $000000000000FFFF ) );
6519: Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word ) : Int64
6520: Function SimpleXMLEncode( const S : string ) : string
6521: Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean )
6522: Function XMLEncode( const S : string ) : string
6523: Function XMLDecode( const S : string ) : string
6524: Function EntityEncode( const S : string ) : string
6525: Function EntityDecode( const S : string ) : string
6526:
6527: procedure RIRegister_CPort_Routines(S: TPSExec);
6528: Procedure EnumComPorts( Ports : TStrings )
6529: Procedure ListComPorts( Ports : TStrings )
6530: Procedure ComPorts( Ports : TStrings ) //Alias to Arduino
6531: Function GetComPorts: TStringlist;
6532: Function StrToBaudRate( Str : string ) : TBAudRate
6533: Function StrToStopBits( Str : string ) : TStopBits
6534: Function StrToDataBits( Str : string ) : TDataBits
6535: Function StrToParity( Str : string ) : TParityBits
6536: Function StrToFlowControl( Str : string ) : TFlowControl
6537: Function BaudRateToStr( BaudRate : TBAudRate ) : string
6538: Function StopBitsToStr( StopBits : TStopBits ) : string
6539: Function DataBitsToStr( DataBits : TDataBits ) : string
6540: Function ParityToStr( Parity : TParityBits ) : string
6541: Function FlowControlToStr( FlowControl : TFlowControl ) : string
6542: Function ComErrorsToStr( Errors : TComErrors ) : String
6543:
6544: Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT ) : BOOL
6545: Function DispatchMessage( const lpMsg : TMsg ) : Longint
6546: Function TranslateMessage( const lpMsg : TMsg ) : BOOL
6547: Function SetMessageQueue( cMessagesMax : Integer ) : BOOL
6548: Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6549: Function GetMessagePos : DWORD
6550: Function GetMessageTime : Longint
6551: Function GetMessageExtraInfo : Longint
6552: Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean ) : string
6553: Procedure JAddToRecentDocs( const Filename : string)
6554: Procedure ClearRecentDocs
6555: Function ExtractIconFromFile( FileName : string; Index : Integer ) : HICON
6556: Function CreateShellLink( const AppName, Desc : string; Dest : string ) : string
6557: Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo )
6558: Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo )
6559: Function RecycleFile( FileToRecycle : string ) : Boolean
6560: Function JCOPYFILE( FromFile, ToDir : string ) : Boolean
6561: Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType ) : UINT
6562: Function ShellObjectTypeConstToEnum( ShellObjectType : UINT ) : TShellObjectType
6563: Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : BOOL
6564: Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : BOOL
6565: Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : BOOL
6566: Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD; dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD;pszGroupName: LP
6567: Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD; dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD;pszGroupName
6568: Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD; dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD; pszGroupName
6569: Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR ) : BOOL
6570:
6571: ***** unit uPSI_JclPeImage;
6572:
6573: Function IsValidPeFile( const FileName : TFileName ) : Boolean
6574: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders ) : Boolean
6575: Function PeCreateNameHintTable( const FileName : TFileName ) : Boolean
6576: Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize : DWORD) : TJclRebaseImageInfo
6577: Function PeVerifyCheckSum( const FileName : TFileName ) : Boolean
6578: Function PeClearCheckSum( const FileName : TFileName ) : Boolean
6579: Function PeUpdateCheckSum( const FileName : TFileName ) : Boolean
6580: Function PeDoesExportFunction(const FileName:TFileName;const FuncName:string;Options:TJclSmartCompOptions):Bool;
6581: Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var ForwardedName : string; Options : TJclSmartCompOptions ) : Boolean
6582: Function PeIsExportFunctionForwarded(const FileName:TFileName;const FunctionName : string; const LibraryName : string; Options : TJclSmartCompOptions):Bool
6583: Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName : string; Options : TJclSmartCompOptions ) : Boolean
6584: Function PeDoesImportLibrary(const FileName:TFileName;const LibraryName:string;Recursive:Boolean):Boolean;
6585: Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive : Boolean; FullPathName : Boolean ) : Boolean
6586: Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const LibraryName:string; IncludeLibNames : Boolean): Boolean
6587: Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings ) : Boolean

```

```

6588: Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings ) : Boolean
6589: Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings ) : Boolean
6590: Function PeResourceKindNames( const FileName : TFileName; ResourceType:TJclPeResourceKind;const
  NamesList:TStrings):Bool
6591: Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings ) : Boolean
6592: Function PeBorDependedPackages( const FileName : TFileName; PackagesList:TStrings;FullPathName,
  Descript:Bool ):Bool;
6593: Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings ) : Boolean;
6594: Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings ) : Boolean;
6595: Function PeCreateRequiredImportList( const FileName : TFileName; RequiredImportsList: TStrings): Boolean;
6596: //Function PeMapImgNtHeaders( const BaseAddress : Pointer ) : PImageNtHeaders
6597: //Function PeMapImgLibraryName( const BaseAddress : Pointer ) : string
6598: //Function PeMapImgSections( const NtHeaders : PImageNtHeaders ) : PImageSectionHeader
6599: //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string ) :
  PImageSectionHeader
6600: //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6601: //Function PeMapImgResolvePackageThunk( Address : Pointer ) : Pointer
6602: Function PeMapFindResource( const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
  __Pointer;
6603: SIRegister_TJclPeSectionStream(CL);
6604: SIRegister_TJclPeMapImgHookItem(CL);
6605: SIRegister_TJclPeMapImgHooks(CL);
6606: //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
  NtHeaders:TImageNtHeaders):Boolean
6607: //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6608: Type TJclBorUmSymbolKind,'(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6609: TJclBorUmSymbolModifier','( smQualified, smLinkProc )
6610: TJclBorUmSymbolModifiers,'( set of TJclBorUmSymbolModifier
6611: TJclBorUmDescription,'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6612: TJclBorUmResult,'( urOk, urNotMangled, urMicrosoft, urError )
6613: TJclPeUmResult,'( umNotMangled, umBorland, umMicrosoft )
6614: Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
  TJclBorUmDescription; var BasePos : Integer ) : TJclBorUmResult;
6615: Function PeBorUnmangleName1( const Name:string;var Unmangled:string;var
  Description:TJclBorUmDescription):TJclBorUmResult;
6616: Function PeBorUnmangleName2( const Name : string; var Unmangled : string ) : TJclBorUmResult;
6617: Function PeBorUnmangleName3( const Name : string ) : string;
6618: Function PeIsNameMangled( const Name : string ) : TJclPeUmResult
6619: Function PeUnmangleName( const Name : string; var Unmangled : string ) : TJclPeUmResult
6620:
6621:
6622: //***** SysTools uPSI_StSystem; *****
6623: Function StCopyFile( const SrcPath, DestPath : AnsiString ) : Cardinal
6624: Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString ) : AnsiString
6625: Function DeleteVolumeLabel( Drive : Char ) : Cardinal
6626: //Procedure EnumerateDirectories(const
  StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6627: //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
  IncludeItem:TIncludeItemFunc);
6628: Function FileHandlesLeft( MaxHandles : Cardinal ) : Cardinal
6629: Function FileMatchesMask( const FileName, FileMode : AnsiString ) : Boolean
6630: Function FileTimeToStDateTime( FileTime : LongInt ) : TStDateTimeRec
6631: Function FindNthSlash( const Path : AnsiString; n : Integer ) : Integer
6632: Function FlushOsBuffers( Handle : Integer ) : Boolean
6633: Function GetCurrentUser : AnsiString
6634: Function GetDiskClass( Drive : Char ) : DiskClass
6635: Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
  SectorsPerCluster:Cardinal):Bool;
6636: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
  DiskSize:Double):Bool;
6637: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
  DiskSize:Comp):Boolean;
6638: { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes }
6639: Function getDiskSpace2(const path: String; index: integer): int64;
6640: Function GetFileCreateDate( const FileName : AnsiString ) : TDateTime
6641: Function StGetFileLastAccess( const FileName : AnsiString ) : TDateTime
6642: Function GetFileLastModify( const FileName : AnsiString ) : TDateTime
6643: Function GetHomeFolder( aForceSlash : Boolean ) : AnsiString
6644: Function GetLongPath( const APath : AnsiString ) : AnsiString
6645: Function GetMachineName : AnsiString
6646: Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType ) : Cardinal
6647: Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean ) : AnsiString
6648: Function GetShortPath( const APath : AnsiString ) : AnsiString
6649: Function GetSystemFolder( aForceSlash : Boolean ) : AnsiString
6650: Function GetTempFolder( aForceSlash : boolean ) : AnsiString
6651: Function StGetWindowsFolder( aForceSlash : boolean ) : AnsiString
6652: Function GetWorkingFolder( aForceSlash : boolean ) : AnsiString
6653: Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer ) : TStDateTimeRec
6654: Function StIsDirectory( const DirName : AnsiString ) : Boolean
6655: Function IsDirectoryEmpty( const S : AnsiString ) : Integer
6656: Function IsDriveReady( Drive : Char ) : Boolean
6657: Function IsFile( const FileName : AnsiString ) : Boolean
6658: Function IsFileArchive( const S : AnsiString ) : Integer
6659: Function IsFileHidden( const S : AnsiString ) : Integer
6660: Function IsFileReadOnly( const S : AnsiString ) : Integer
6661: Function IsFileSystem( const S : AnsiString ) : Integer
6662: Function LocalDateTimeToGlobal( const DTL : TStDateTimeRec; MinOffset : Integer ) : TStDateTimeRec
6663: Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char ) : Cardinal
6664: Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer ) : Boolean

```

```

6665: Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType ) : Cardinal
6666: Procedure SplitPath( const APath : AnsiString; Parts : TStrings )
6667: Function StDateTimeToFileTime( const FileTime : TStDateTimeRec ) : LongInt
6668: Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec ) : Longint
6669: Function UnixTimeToStDateTime( UnixTime : Longint ) : TStDateTimeRec
6670: Function ValidDrive( Drive : Char ) : Boolean
6671: Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char ) : Cardinal
6672:
6673: //*****unit uPSI_JcLANMan;*****
6674: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
  const PasswordNeverExpires : Boolean ) : Boolean
6675: Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
  const PasswordNeverExpires : Boolean ) : Boolean
6676: Function DeleteAccount( const Servername, Username : string ) : Boolean
6677: Function DeleteLocalAccount( Username : string ) : Boolean
6678: Function CreateLocalGroup( const Server, Groupname, Description : string ) : Boolean
6679: Function CreateGlobalGroup( const Server, Groupname, Description : string ) : Boolean
6680: Function DeleteLocalGroup( const Server, Groupname : string ) : Boolean
6681: Function GetLocalGroups( const Server : string; const Groups : TStrings ) : Boolean
6682: Function GetGlobalGroups( const Server : string; const Groups : TStrings ) : Boolean
6683: Function LocalGroupExists( const Group : string ) : Boolean
6684: Function GlobalGroupExists( const Server, Group : string ) : Boolean
6685: Function AddAccountToLocalGroup( const Accountname, Groupname : string ) : Boolean
6686: Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID ) : string
6687: Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string )
6688: Function IsLocalAccount( const AccountName : string ) : Boolean
6689: Function TimeStampInterval( StartStamp, EndStamp : TDateTime ) : integer
6690: Function GetRandomString( NumChar : cardinal ) : string
6691:
6692: //*****unit uPSI_cUtils;*****
6693: Types('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )'
6694: Function cIsWinNT : boolean
6695: Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
  Multitasking:Boolean;
6696: Function cExecutefile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ) : THandle
6697: Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
  CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6698: Function cGetShortName( FileName : string ) : string
6699: Procedure cShowError( Msg : String )
6700: Function cCommaStrToStr( s : string; formatstr : string ) : string
6701: Function cIncludeQuoteIfSpaces( s : string ) : string
6702: Function cIncludeQuoteIfNeeded( s : string ) : string
6703: Procedure cLoadFilefromResource( const FileName : string )
6704: Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6705: Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET ) : boolean;
6706: Function cBuildFilter1( var value : string; const _filters : array of string ) : boolean;
6707: Function cCodeInstoStr( s : string ) : string
6708: Function cStrtoCodeIns( s : string ) : string
6709: Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6710: Function cAttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6711: Procedure cStrToPoint( var pt : TPoint; value : string )
6712: Function cPointtoStr( const pt : TPoint ) : string
6713: Function cListtoStr( const List : TStrings ) : string
6714: Function ListtoStr( const List : TStrings ) : string
6715: Procedure StrtoList( s : string; const List : TStrings; const delimiter : char )
6716: Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char )
6717: Function cGetFileType( const FileName : string ) : TUnitType
6718: Function cGetExTyp( const FileName : string ) : TExUnitType
6719: Procedure cSetPath( Add : string; const UseOriginal : boolean )
6720: Function cExpandFileto( const FileName : string; const basePath : string ) : string
6721: Function cFileSamePath( const FileName : string; const TestPath : string ) : boolean
6722: Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem )
6723: Function cGetLastPos( const SubStr : string; const S : string ) : integer
6724: Function cGenMakePath( FileName : String ) : String;
6725: Function cGenMakePath2( FileName : String ) : String
6726: Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean ) : String;
6727: Function cGetRealPath( BrokenFileName : String; Directory : String ) : String
6728: Function cCalcMod( Count : Integer ) : Integer
6729: Function cGetVersionString( FileName : string ) : string
6730: Function cCheckChangeDir( var Dir : string ) : boolean
6731: Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6732: Function cIsNumeric( s : string ) : boolean
6733: Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6734: Function AttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6735: Function GetfileTyp( const FileName : string ) : TUnitType
6736: Function Atoi(const aStr: string): integer
6737: Function Itoa(const aint: integer): string
6738: Function Atof(const aStr: string): double';
6739: Function Atol(const aStr: string): longint';
6740:
6741:
6742: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6743: begin
6744:   FindClass('TOBJECT'), 'EHTTP
6745:   FindClass('TOBJECT'), 'EHTTPParser
6746:   //AnsiCharSet', 'set of AnsiChar
6747:   AnsiStringArray', 'array of AnsiString
6748:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6749:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )

```

```

6750: THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6751: +'TPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6752: +'CustomMinVersion : Integer; end
6753: THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6754: +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6755: +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6756: +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6757: +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6758: +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6759: +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6760: +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSince'
6761: +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6762: +'nection, hntOrigin, hntKeepAlive )'
6763: THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6764: THTTPCustomHeader', 'record FieldName : AnsiString;FieldValue :'
6765: +' AnsiString; end
6766: //THTTPCustomHeader', '^THTTPCustomHeader // will not work
6767: THTTPContentLengthEnum', '( hcNone, hcByteCount )'
6768: THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6769: //THTTPContentLength', '^THTTPContentLength // will not work
6770: THTTPContentTypeMajor', '( hctCustom, hctText, hctImage )'
6771: THTTPContentTypeEnum', '( hcNone, hctCustomParts, hctCustomStr'
6772: +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6773: +'xCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6774: +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctAppli'
6775: +'ctionCustom, hctAudioCustom, hctVideoCustom )'
6776: THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6777: +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray; '
6778: +'CustomStr : AnsiString; end
6779: THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )'
6780: THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6781: +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6782: +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6783: +'String; DateTime : TDateTime; Custom : AnsiString; end
6784: THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )'
6785: THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnum'
6786: +'m; Custom : AnsiString; end
6787: THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )'
6788: THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum; '
6789: +'Custom : AnsiString; end
6790: THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )'
6791: THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6792: THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )'
6793: THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6794: +' , hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )'
6795: THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6796: +' , hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6797: +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )'
6798: THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6799: THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6800: +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )'
6801: THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end'
6802: THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )'
6803: THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6804: +'FieldEnum; List : array of THTTPContentEncoding; end
6805: THTTPRetryAfterFieldEnum', '( harfNone, harfCustom, harfDate, harfSeconds )'
6806: THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum; '
6807: +'Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6808: THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )'
6809: THTTPContentRangeField', 'record Value : THTTPContentRangeField'
6810: +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6811: THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )'
6812: THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6813: THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField'
6814: THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6815: +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6816: +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6817: +'CustomFieldArray; Custom : AnsiString; end
6818: //THTTPSetCookieField', '^THTTPSetCookieField // will not work
6819: THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6820: THTTPCookiefieldEnum', '( hcoNone, hcoDecoded, hcoCustom )'
6821: THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6822: //THTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6823: THTTPCookiefieldEntryArray', 'array of THTTPCookieFieldEntry
6824: THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6825: +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6826: THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6827: +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength; '
6828: +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6829: +' ; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6830: THTTPCustomHeaders', 'array of THTTPCustomHeader
6831: //THTTPFixedHeaders', 'array[THTTPHeaderNameEnum] of AnsiString
6832: THTTPFixedHeaders', 'array[0..42] of AnsiString
6833: THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6834: +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )'
6835: THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6836: THTTPRequestStartLine', 'record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6837: THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders; '
6838: +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'

```

```

6839: +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6840: //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6841: THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6842: +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6843: THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6844: THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6845: +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6846: THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6847: +' ; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6848: +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified : '
6849: +' THTTPDateField; Age : THTTPAgeField; end
6850: //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6851: THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6852: +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6853: Function HTTPMessageHasContent( const H : THTTPCommonHeaders ) : Boolean
6854: Procedure InitHTTPRequest( var A : THTTPRequest )
6855: Procedure InitHTTPResponse( var A : THTTPResponse )
6856: Procedure ClearHTTPVersion( var A : THTTPVersion )
6857: Procedure ClearHTTPContentLength( var A : THTTPContentLength )
6858: Procedure ClearHTTPContentType( var A : THTTPContentType )
6859: Procedure ClearHTTPDateField( var A : THTTPDateField )
6860: Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding )
6861: Procedure ClearHTTPConnectionField( var A : THTTPConnectionField )
6862: Procedure ClearHTTPAgeField( var A : THTTPAgeField )
6863: Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding )
6864: Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField )
6865: Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField )
6866: Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField )
6867: Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders )
6868: //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders )
6869: Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders )
6870: Procedure ClearHTTPCookieField( var A : THTTPCookieField )
6871: Procedure ClearHTTPMethod( var A : THTTPMethod )
6872: Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine )
6873: Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader )
6874: Procedure ClearHTTPRequest( var A : THTTPRequest )
6875: Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine )
6876: Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader )
6877: Procedure ClearHTTPResponse( var A : THTTPResponse )
6878: THTTPStringOption', '( hsoNone )
6879: THTTPStringOptions', 'set of THTTPStringOption
6880: FindClass('TOBJECT'), 'TansiStringBuilder
6881:
6882: Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TansiStringBuilder; P:THTTPStringOptions;
6883: Procedure BuildStrHTTPContentLengthValue(const
6884: A:THTTPContentLength;B:TansiStringBuilder;P:THTTPStringOptions)
6885: Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
6886: B:TansiStringBuilder;P:THTTPStringOptions)
6887: Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TansiStringBuilder;const
6888: P:THTTPStringOptions)
6889: Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TansiStringBuilder; const
6890: P:THTTPStringOptions)
6891: Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
6892: B : TansiStringBuilder; const P : THTTPStringOptions)
6893: Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TansiStringBuilder; const P :
6894: THTTPStringOptions)
6895: Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TansiStringBuilder;const
6896: P:THTTPStringOptions);
6897: Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
6898: TansiStringBuilder; const P : THTTPStringOptions)
6899: Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TansiStringBuilder;
6900: const P : THTTPStringOptions)
6901: Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
6902: B:TansiStringBuilder;const P:THTTPStringOptions)
6903: Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TansiStringBuilder;
6904: const P : THTTPStringOptions)
6905: Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TansiStringBuilder; const P
6906: : THTTPStringOptions)
6907: Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TansiStrBuilder;const
6908: P:THTTPStringOptions)
6909: Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TansiStringBuilder; const P
6910: : THTTPStringOptions)
6911: Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TansiStringBuilder;
6912: const P : THTTPStringOptions)
6913: Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TansiStringBuilder; const P
6914: : THTTPStringOptions)
6915: Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TansiStringBuilder; const P :
6916: THTTPStringOptions)

```

```

6906: Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
6907: const P : THTTPStringOptions)
6908: Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
6909: P:THTTPStringOptions);
6910: Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
6911: TAnsiStringBuilder; const P : THTTPStringOptions)
6912: Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
6913: THTTPStrOptions);
6914: Procedure BuildStrHTTPHeader(const A : THTTPContentType) : AnsiString
6915: Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6916: Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6917: Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6918: Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6919: Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6920: Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
6921: Domain:AnsiString;const Secure:Boolean; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6922: +PHeaderNameEnum; const HeaderPtr : __Pointer) : Boolean
6923: SIRegister_THTTPParser(CL);
6924: FindClass ('TOBJECT'), 'THTTPContentDecoder
6925: THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6926: THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6927: THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,
6928: +' crcsContentCRLF, crcsTrailer, crcsFinished )
6929: THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6930: SIRegister_THTTPContentDecoder(CL);
6931: THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6932: FindClass ('TOBJECT'), 'THTTPContentReader
6933: THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6934: THTTPContentReaderLogEvent', 'Procedure (const Sender:THTTPContentReader;const LogMsg:String;const
6935: LogLevel:Int;
6936: SIRegister_THTTPContentReader(CL);
6937: THTTPContentWriterMechanism', '( hctmEvent, hctmString, hctmStream, hctmFile )
6938: FindClass ('TOBJECT'), 'THTTPContentWriter
6939: THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6940: SIRegister_THTTPContentWriter(CL);
6941: Procedure SelfTestcHTTPUtil
6942: begin
6943:   (*-----*)
6944:   'TLSLibraryVersion', 'String '1.00
6945:   'TLSerror_None', 'LongInt'( 0 );
6946:   'TLSerror_InvalidBuffer', 'LongInt'( 1 );
6947:   'TLSerror_InvalidParameter', 'LongInt'( 2 );
6948:   'TLSerror_InvalidCertificate', 'LongInt'( 3 );
6949:   'TLSerror_InvalidState', 'LongInt'( 4 );
6950:   'TLSerror_DecodeError', 'LongInt'( 5 );
6951:   'TLSerror_BadProtocol', 'LongInt'( 6 );
6952:   Function TLSErrorMessage( const TLSError : Integer ) : String
6953:   SIRegister_ETLSError(CL);
6954:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6955:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6956:   Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6957:   Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6958:   Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6959:   Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6960:   Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion ) : Boolean
6961:   Function IsSSL2( const A : TTLSProtocolVersion ) : Boolean
6962:   Function IsSSL3( const A : TTLSProtocolVersion ) : Boolean
6963:   Function IsTLS10( const A : TTLSProtocolVersion ) : Boolean
6964:   Function IsTLS11( const A : TTLSProtocolVersion ) : Boolean
6965:   Function IsTLS12( const A : TTLSProtocolVersion ) : Boolean
6966:   Function IsTLS10OrLater( const A : TTLSProtocolVersion ) : Boolean
6967:   Function IsTLS11OrLater( const A : TTLSProtocolVersion ) : Boolean
6968:   Function IsTLS12OrLater( const A : TTLSProtocolVersion ) : Boolean
6969:   Function IsFutureTLSVersion( const A : TTLSProtocolVersion ) : Boolean
6970:   Function IsKnownTLSVersion( const A : TTLSProtocolVersion ) : Boolean
6971:   Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion ) : String
6972:   Function TLSProtocolVersionName( const A : TTLSProtocolVersion ) : String
6973:   PTLSRandom', '^TTLSSRandom // will not work
6974:   Procedure InitTLSRandom( var Random : TTLSSRandom )
6975:   Function TLSRandomToStr( const Random : TTLSSRandom ) : AnsiString
6976:   'TLSSessionIDMaxLen', 'LongInt'( 32 );
6977:   Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString )
6978:   Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6979:   Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6980:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSSignatureAlgorithm';
6981:   +' ; Signature : TTLSSignatureAlgorithm; end
6982: // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6983:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6984:   TTLSSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_
6985:   +' DSS, tlskeadHE_RSA, tlskeadH_Anon, tlskeadH_DSS, tlskeadH_RSA )

```

```

6986: TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6987: +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )'
6988: TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : Integer;
6989: +'integer; Supported : Boolean; end'
6990: PTLSMacAlgorithmInfo', '^TTLSSMacAlgorithmInfo // will not work'
6991: 'TLS_MAC_MAXDIGESTSIZE', 'LongInt'( 64 );
6992: TTLSPRFAlgorithm', '( tlspaSHA256 )
6993: Function tlsp_MD5( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6994: Function tlsp_SHA1( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6995: Function tlsp_SHA256( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6996: Function tlsp_SHA512( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6997: Function tlsp1OPRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer ) : AnsiString
6998: Function tlsp12PRF_SHA256( const Secret, ALLabel, Seed : AnsiString; const Size : Integer ) : AnsiString
6999: Function tlsp12PRF_SHA512( const Secret, ALLabel, Seed : AnsiString; const Size : Integer ) : AnsiString
7000: Function TLSPRF( const ProtoVersion:TTLSSProtocolVersion;const Secret,ALabel,Seed:AString;const
    Size:Int ):AString;
7001: Function tlsp10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
    Size:Integer):AnsiString
7002: Function tlsp12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
    Size:Int):AnsiString;
7003: Function tlsp12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
    Size:Int):AnsiString;
7004: Function TLSKeyBlock(const ProtocolVersion:TTLSSProtocolVersion; const MasterSecret, ServerRandom,
    ClientRandom : AnsiString; const Size : Integer ) : AnsiString
7005: Function tlsp10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
7006: Function tlsp12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
7007: Function tlsp12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
7008: Function TLSMasterSecret( const ProtocolVersion: TTLSSProtocolVersion;const PreMasterSecret,ClientRandom,
    ServerRandom:AnsiString ) : AnsiString
7009: 'TLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr
    +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
    +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end'
7010: Procedure GenerateTLSKeys( const ProtocolVersion : TTLSSProtocolVersion; const MACKeyBits, CipherKeyBits,
    IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TLSKeys )
7011: Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSSProtocolVersion;const IsExportable:Boolean;const
    ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TLSKeys)
7012: 'TLS_PLAINTEXT_FRAGMENT_MAXSIZE', 'LongInt'( 16384 - 1 );
7013: 'TLS_COMPRESSED_FRAGMENT_MAXSIZE', 'LongInt'( 16384 + 1024 );
7014: Procedure SelfTestcTLSUtils
7015: end;
7016: 
7017: (*-----*)
7018: 
7019: 
7020: procedure SIRegister_Reversi(CL: TPSPPascalCompiler);
7021: begin
7022:   sPosData', 'record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
    integer; disks : integer; mx : integer; my : integer; end
7023: // pBoard', '^pBoard // will not work
7024: Function rCalculateData( cc : byte; cx, cy : integer ) : sPosData
7025: Function rCheckMove( color : byte; cx, cy : integer ) : integer
7026: //Function rDoStep( data : pBoard ) : word
7027: Function winExecAndWait( const sAppPath : string; wVisible : word ) : boolean
7028: end;
7029: 
7030: procedure SIRegister_IWDBCommon(CL: TPSPPascalCompiler);
7031: begin
7032:   Function InEditMode( ADataSet : TDataSet ) : Boolean
7033:   Function CheckDataSource( ADataSource : TDataSource ) : Boolean;
7034:   Function CheckDataSource1( ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
7035:   Function GetFieldText( AField : TField ) : String
7036: end;
7037: 
7038: procedure SIRegister_SortGrid(CL: TPSPPascalCompiler);
7039: begin
7040:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7041:   TMyPrintRange', '( prAll, prSelected )
7042:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten
    +ded, ssDateTime, ssTime, ssCustom )
7043:   TSortDirection', '( sdAscending, sdDescending )
7044:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7045:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer
    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7046:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7047:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7048:   SIRegister_TSortOptions(CL);
7049:   SIRegister_TPrintOptions(CL);
7050:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7051:   SIRegister_TSortedList(CL);
7052:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7053:   SIRegister_TSortedList(CL);
7054:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7055:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7056:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7057:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment
    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7058:   SIRegister_TFontSetting(CL);
7059:   SIRegister_TFontList(CL);
7060:   AddTypes(TFormatDrawCellEvent)', 'Procedure ( Sender : TObject; Col, Row :
    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7061:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7062:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7063:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)

```

```

7066: TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7067: TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7068: TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7069: TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : integer
7070:   +'r; var SortStyle : TSortStyle)
7071: TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :
7072:   +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7073: SIRegister_TSORTGrid(CL);
7074: Function ExtendedCompare( const Str1, Str2 : String) : Integer
7075: Function NormalCompare( const Str1, Str2 : String) : Integer
7076: Function DateTimeCompare( const Str1, Str2 : String) : Integer
7077: Function NumericCompare( const Str1, Str2 : String) : Integer
7078: Function TimeCompare( const Str1, Str2 : String) : Integer
7079: //Function Compare( Item1, Item2 : Pointer) : Integer
7080: end;
7081;
7082: ***** procedure Register_IB(CL: TPPascalCompiler);
7083: Procedure IBAalloc( var P, OldSize, NewSize : Integer)
7084: Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
7085: Procedure IBDATABaseError
7086: Function StatusVector : PISC_STATUS
7087: Function StatusVectorArray : PStatusVector
7088: Function CheckStatusVector( ErrorCode : array of ISC_STATUS) : Boolean
7089: Function StatusVectorAsText : string
7090: Procedure SetIBDATABaseErrorMessages( Value : TIBDATABaseErrorMessages)
7091: Function GetIBDATABaseErrorMessages : TIBDATABaseErrorMessages
7092:
7093:
7094: //*****unit uPSI_BoldUtils;*****
7095: Function CharCount( c : char; const s : string) : integer
7096: Function BoldNamesEqual( const name1, name2 : string) : Boolean
7097: Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7098: Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7099: Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7100: Function BoldTrim( const S : string) : string
7101: Function BoldIsPrefix( const S, Prefix : string) : Boolean
7102: Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7103: Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7104: Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7105: Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7106: Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7107: //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7108: Function CapitalisedToSpaced( Capitalised : String) : String
7109: Function SpacedToCapitalised( Spaced : String) : String
7110: Function BooleanToString( BoolValue : Boolean) : String
7111: Function StringToBoolean( StrValue : String) : Boolean
7112: Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7113: Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7114: Function StringListToVarArray( List : TStringList) : variant
7115: Function IsLocalMachine( const Machinename : WideString) : Boolean
7116: Function GetComputerNameStr : string
7117: Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7118: Function BoldStrToDateFmt( const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7119: Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7120: Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7121: Function BoldParseFormattedDate(const value:String;const formats:array of string; var
Date:TDateTime):Boolean;
7122: Procedure EnsureTrailing( var Str : String; ch : char)
7123: Function BoldDirectoryExists( const Name : string) : Boolean
7124: Function BoldForceDirectories( Dir : string) : Boolean
7125: Function BoldRootRegistryKey : string
7126: Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7127: Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7128: Function LogicalAnd( A, B : Integer) : Boolean
7129: record TByHandleFileInformation dwFileAttributes : DWORD;
7130:   +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7131:   +' : TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSize'
7132:   +'eLow : DWORD; nNumberOfLinks : DWORD; nfileIndexHigh : DWORD; nfileIndexLow : DWORD; end
7133: Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7134: Function IsFirstInstance : Boolean
7135: Procedure ActivateFirst( AString : PChar)
7136: Procedure ActivateFirstCommandLine
7137: function MakeAckPkt(const BlockNumber: Word): string;
7138: procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7139: procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7140: procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer; E: Exception); overload;
7141: procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7142: function IdStrToWord(const Value: String): Word;
7143: function IdWordToStr(const Value: Word): WordStr;
7144: Function HasInstructionSet( const InstructionSet : TCPUIInstructionSet) : Boolean
7145: Function CPUFeatures : TCPUIFeatures
7146:
7147: procedure SIRegister_xrtl_util_CPUUtils(CL: TPPascalCompiler);
7148: begin
7149:   AddTypeS('TXRTLBitIndex', 'Integer'
7150:   Function XRTLswapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7151:   Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7152:   Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7153:   Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal

```

```

7154: Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex ) : Cardinal
7155: Function XRTLSwapHiLo16( X : Word ) : Word
7156: Function XRTLSwapHiLo32( X : Cardinal ) : Cardinal
7157: Function XRTLSwapHiLo64( X : Int64 ) : Int64
7158: Function XRTLROL32( A, S : Cardinal ) : Cardinal
7159: Function XRTLROL32( A, S : Cardinal ) : Cardinal
7160: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7161: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7162: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7163: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7164: //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer )
7165: //Procedure XRTLIncBlock( P : PByteArray; Len : integer )
7166: Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer )
7167: Function XRTLPopulation( A : Cardinal ) : Cardinal
7168: end;
7169:
7170: Function XRTLURLDecode( const ASrc : WideString ) : WideString
7171: Function XRTLURLEncode( const ASrc : WideString ) : WideString
7172: Function XRTLURINormalize( const AURI : WideString ) : WideString
7173: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,VPassword : WideString)
7174: Function XRTLExtractLongPathName(APath: string): string;
7175:
7176: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7177: begin
7178:   Int8', 'ShortInt
7179:   AddTypeS('Int16', 'SmallInt
7180:   Int32', 'LongInt
7181:   UInt8', 'Byte
7182:   UInt16', 'Word
7183:   UInt32', 'LongWord
7184:   UInt64', 'Int64
7185:   Word8', 'UInt8
7186:   Word16', 'UInt16
7187:   Word32', 'UInt32
7188:   Word64', 'UInt64
7189:   LargeInt', 'Int64
7190:   NativeInt', 'Integer
7191:   AddTypes('NativeUInt', 'Cardinal
7192:   Const('BitsPerByte','LongInt'( 8);
7193:   Const('BitsPerWord','LongInt'( 16);
7194:   Const('BitsPerLongWord','LongInt'( 32);
7195:   //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7196:   //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7197:   Function MinI( const A, B : Integer ) : Integer
7198:   Function MaxI( const A, B : Integer ) : Integer
7199:   Function MinC( const A, B : Cardinal ) : Cardinal
7200:   Function MaxC( const A, B : Cardinal ) : Cardinal
7201:   Function SumClipI( const A, I : Integer ) : Integer
7202:   Function SumClipC( const A : Cardinal; const I : Integer ) : Cardinal
7203:   Function InByteRange( const A : Int64 ) : Boolean
7204:   Function InWordRange( const A : Int64 ) : Boolean
7205:   Function InLongWordRange( const A : Int64 ) : Boolean
7206:   Function InShortIntRange( const A : Int64 ) : Boolean
7207:   Function InSmallIntRange( const A : Int64 ) : Boolean
7208:   Function InLongIntRange( const A : Int64 ) : Boolean
7209:   AddTypeS('Bool8', 'ByteBool
7210:   AddTypeS('Bool16', 'WordBool
7211:   AddTypeS('Bool32', 'LongBool
7212:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7213:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7214:   Function ReverseCompareResult( const C : TCompareResult ) : TCompareResult
7215:   Const('MinSingle','Single').setExtended( 1.5E-45 );
7216:   Const('MaxSingle','Single').setExtended( 3.4E+38 );
7217:   Const('MinDouble','Double').setExtended( 5.0E-324 );
7218:   Const('MaxDouble','Double').setExtended( 1.7E+308 );
7219:   Const('MinExtended','Extended').setExtended(3.4E-4932);
7220:   Const('MaxExtended','Extended').setExtended(1.1E+4932);
7221:   Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807 );
7222:   Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807 );
7223:   Function MinF( const A, B : Float ) : Float
7224:   Function MaxF( const A, B : Float ) : Float
7225:   Function ClipF( const Value : Float; const Low, High : Float ) : Float
7226:   Function InSingleRange( const A : Float ) : Boolean
7227:   Function InDoubleRange( const A : Float ) : Boolean
7228:   Function InCurrencyRange( const A : Float ) : Boolean;
7229:   Function InCurrencyRangel( const A : Int64 ) : Boolean;
7230:   Function FloatExponentBase2( const A : Extended; var Exponent : Integer ) : Boolean
7231:   Function FloatExponentBase10( const A : Extended; var Exponent : Integer ) : Boolean
7232:   Function FloatIsInfinity( const A : Extended ) : Boolean
7233:   Function FloatIsNaN( const A : Extended ) : Boolean
7234:   Const('SingleCompareDelta','Extended').setExtended( 1.0E-34 );
7235:   Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280 );
7236:   Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400 );
7237:   Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34 );
7238:   Function FloatZero( const A : Float; const CompareDelta : Float ) : Boolean
7239:   Function FloatOne( const A : Float; const CompareDelta : Float ) : Boolean
7240:   Function FloatsEqual( const A, B : Float; const CompareDelta : Float ) : Boolean
7241:   Function FloatsCompare( const A, B : Float; const CompareDelta : Float ) : TCompareResult

```

```

7242: Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7243: Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7244: Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7245: Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7246: Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7247: Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7248: Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7249: Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7250: Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7251: Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7252: Function cIsHighBitSet( const Value : LongWord) : Boolean
7253: Function SetBitScanForward( const Value : LongWord) : Integer;
7254: Function SetBitScanForward1( const Value : LongWord) : Integer;
7255: Function SetBitScanReverse( const Value : LongWord) : Integer;
7256: Function SetBitScanReverse1( const Value : LongWord) : Integer;
7257: Function ClearBitScanForward( const Value : LongWord) : Integer;
7258: Function ClearBitScanForward1( const Value : LongWord) : Integer;
7259: Function ClearBitScanReverse( const Value : LongWord) : Integer;
7260: Function ClearBitScanReverse1( const Value : LongWord) : Integer;
7261: Function cReverseBits( const Value : LongWord) : LongWord;
7262: Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7263: Function cSwapEndian( const Value : LongWord) : LongWord
7264: Function cTwosComplement( const Value : LongWord) : LongWord
7265: Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7266: Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7267: Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7268: Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7269: Function cBitCount( const Value : LongWord) : LongWord
7270: Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7271: Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7272: Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7273: Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7274: Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7275: Function ClearBitRange(const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7276: Function ToggleBitRange(const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7277: Function IsBitRangeSet(const Value: LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7278: Function IsBitRangeClear(const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : Boolean
7279: // AddTypeS('CharSet', 'set of AnsiChar'
7280: AddTypeS('CharSet', 'set of Char'           //!!!
7281: AddTypeS('AnsiCharSet', 'TCharSet'
7282: AddTypeS('ByteSet', 'set of Byte'
7283: AddTypeS('AnsiChar', 'Char
7284: // Function AsCharSet( const C : array of AnsiChar) : CharSet
7285: Function AsByteSet( const C : array of Byte) : ByteSet
7286: Procedure ComplementChar( var C : CharSet; const Ch : Char)
7287: Procedure ClearCharSet( var C : CharSet)
7288: Procedure FillCharSet( var C : CharSet)
7289: procedure FillCharSearchRec; // with 0
7290: Procedure ComplementCharSet( var C : CharSet)
7291: Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7292: Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7293: Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7294: Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7295: Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7296: Function IsSubSet( const A, B : CharSet) : Boolean
7297: Function IsEqual( const A, B : CharSet) : Boolean
7298: Function IsEmpty( const C : CharSet) : Boolean
7299: Function IsComplete( const C : CharSet) : Boolean
7300: Function cCharCount( const C : CharSet) : Integer
7301: Procedure ConvertCaseInsensitive( var C : CharSet)
7302: Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7303: Function IntRangeLength( const Low, High : Integer) : Int64
7304: Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7305: Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7306: Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7307: Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7308: Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7309: Function CardRangeLength( const Low, High : Cardinal) : Int64
7310: Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7311: Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7312: Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7313: Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7314: Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7315: AddTypeS('UnicodeChar', 'WideChar
7316: Function Compare( const I1, I2 : Boolean) : TCompareResult;
7317: Function CompareI( const I1, I2 : Integer) : TCompareResult;
7318: Function Compare2( const I1, I2 : Int64) : TCompareResult;
7319: Function Compare3( const I1, I2 : Extended) : TCompareResult;
7320: Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7321: Function CompareW( const I1, I2 : WideString) : TCompareResult
7322: Function cSgn( const A : LongInt) : Integer;
7323: Function cSgn1( const A : Int64) : Integer;
7324: Function cSgn2( const A : Extended) : Integer;
7325: AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7326: Function AnsiCharToInt( const A : AnsiChar) : Integer
7327: Function WideCharToInt( const A : WideChar) : Integer
7328: Function CharToInt( const A : Char) : Integer
7329: Function IntToAnsiChar( const A : Integer) : AnsiChar
7330: Function IntToWideChar( const A : Integer) : WideChar

```

```

7331: Function IntToChar( const A : Integer ) : Char
7332: Function IsHexAnsiChar( const Ch : AnsiChar ) : Boolean
7333: Function IsHexWideChar( const Ch : WideChar ) : Boolean
7334: Function IsHexChar( const Ch : Char ) : Boolean
7335: Function HexAnsiCharToInt( const A : AnsiChar ) : Integer
7336: Function HexWideCharToInt( const A : WideChar ) : Integer
7337: Function HexCharToInt( const A : Char ) : Integer
7338: Function IntToUpperHexAnsiChar( const A : Integer ) : AnsiChar
7339: Function IntToUpperHexWideChar( const A : Integer ) : WideChar
7340: Function IntToUpperHexChar( const A : Integer ) : Char
7341: Function IntToLowerHexAnsiChar( const A : Integer ) : AnsiChar
7342: Function IntToLowerHexWideChar( const A : Integer ) : WideChar
7343: Function IntToLowerHexChar( const A : Integer ) : Char
7344: Function IntToStringA( const A : Int64 ) : AnsiString
7345: Function IntToStringW( const A : Int64 ) : WideString
7346: Function IntToString( const A : Int64 ) : String
7347: Function UIntToStringA( const A : NativeUInt ) : AnsiString
7348: Function UIntToStringW( const A : NativeUInt ) : WideString
7349: Function UIntToString( const A : NativeUInt ) : String
7350: Function LongWordToStrA( const A : LongWord; const Digits : Integer ) : AnsiString
7351: Function LongWordToStrW( const A : LongWord; const Digits : Integer ) : WideString
7352: Function LongWordToStrU( const A : LongWord; const Digits : Integer ) : UnicodeString
7353: Function LongWordToStr( const A : LongWord; const Digits : Integer ) : String
7354: Function LongWordToHexA( const A : LongWord; const Digits : Integer; const UpperCase : Boolean ) : AnsiString
7355: Function LongWordToHexW( const A : LongWord; const Digits : Integer; const UpperCase : Boolean ) : WideString
7356: Function LongWordToHex( const A : LongWord; const Digits : Integer; const UpperCase : Boolean ) : String
7357: Function LongWordToOctA( const A : LongWord; const Digits : Integer ) : AnsiString
7358: Function LongWordToOctW( const A : LongWord; const Digits : Integer ) : WideString
7359: Function LongWordToOct( const A : LongWord; const Digits : Integer ) : String
7360: Function LongWordToBinA( const A : LongWord; const Digits : Integer ) : AnsiString
7361: Function LongWordToBinW( const A : LongWord; const Digits : Integer ) : WideString
7362: Function LongWordToBin( const A : LongWord; const Digits : Integer ) : String
7363: Function TryStringToInt64A( const S : AnsiString; out A : Int64 ) : Boolean
7364: Function TryStringToInt64W( const S : WideString; out A : Int64 ) : Boolean
7365: Function TryStringToInt64( const S : String; out A : Int64 ) : Boolean
7366: Function StringToInt64DefA( const S : AnsiString; const Default : Int64 ) : Int64
7367: Function StringToInt64DefW( const S : WideString; const Default : Int64 ) : Int64
7368: Function StringToInt64Def( const S : String; const Default : Int64 ) : Int64
7369: Function StringToInt64A( const S : AnsiString ) : Int64
7370: Function StringToInt64W( const S : WideString ) : Int64
7371: Function StringToInt64( const S : String ) : Int64
7372: Function TryStringToIntA( const S : AnsiString; out A : Integer ) : Boolean
7373: Function TryStringToIntW( const S : WideString; out A : Integer ) : Boolean
7374: Function TryStringToInt( const S : String; out A : Integer ) : Boolean
7375: Function StringToIntDefA( const S : AnsiString; const Default : Integer ) : Integer
7376: Function StringToIntDefW( const S : WideString; const Default : Integer ) : Integer
7377: Function StringToIntDef( const S : String; const Default : Integer ) : Integer
7378: Function StringToIntA( const S : AnsiString ) : Integer
7379: Function StringToIntW( const S : WideString ) : Integer
7380: Function StringToInt( const S : String ) : Integer
7381: Function TryStringToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7382: Function TryStringToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7383: Function TryStringToLongWord( const S : String; out A : LongWord ) : Boolean
7384: Function StringToLongWordA( const S : AnsiString ) : LongWord
7385: Function StringToLongWordW( const S : WideString ) : LongWord
7386: Function StringToLongWord( const S : String ) : LongWord
7387: Function HexToIntA( const S : AnsiString ) : NativeUInt
7388: Function HexToIntW( const S : WideString ) : NativeUInt
7389: Function HexToInt( const S : String ) : NativeUInt
7390: Function TryHexToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7391: Function TryHexToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7392: Function TryHexToLongWord( const S : String; out A : LongWord ) : Boolean
7393: Function HexToLongWordA( const S : AnsiString ) : LongWord
7394: Function HexToLongWordW( const S : WideString ) : LongWord
7395: Function HexToLongWord( const S : String ) : LongWord
7396: Function TryOctToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7397: Function TryOctToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7398: Function TryOctToLongWord( const S : String; out A : LongWord ) : Boolean
7399: Function OctToLongWordA( const S : AnsiString ) : LongWord
7400: Function OctToLongWordW( const S : WideString ) : LongWord
7401: Function OctToLongWord( const S : String ) : LongWord
7402: Function TryBinToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7403: Function TryBinToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7404: Function TryBinToLongWord( const S : String; out A : LongWord ) : Boolean
7405: Function BinToLongWordA( const S : AnsiString ) : LongWord
7406: Function BinToLongWordW( const S : WideString ) : LongWord
7407: Function BinToLongWord( const S : String ) : LongWord
7408: Function FloatToStringA( const A : Extended ) : AnsiString
7409: Function FloatToStringW( const A : Extended ) : WideString
7410: Function FloatToString( const A : Extended ) : String
7411: Function TryStringToFloatA( const A : AnsiString; out B : Extended ) : Boolean
7412: Function TryStringToFloatW( const A : WideString; out B : Extended ) : Boolean
7413: Function TryStringToFloat( const A : String; out B : Extended ) : Boolean
7414: Function StringToFloatA( const A : AnsiString ) : Extended
7415: Function StringToFloatW( const A : WideString ) : Extended
7416: Function StringToFloat( const A : String ) : Extended
7417: Function StringToFloatDefA( const A : AnsiString; const Default : Extended ) : Extended
7418: Function StringToFloatDefW( const A : WideString; const Default : Extended ) : Extended
7419: Function StringToFloatDef( const A : String; const Default : Extended ) : Extended

```

```

7420: Function EncodeBase64( const S:Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
    PadChar: AnsiChar ) : AnsiString
7421: Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet ) : AnsiString
7422: unit uPSI_cfundamentUtils;
7423: Const('b64_MIMEBase64','Str').String('ABCDEFIGHIJKLNMOPQRSTUVWXYZZabedefghijklnopqrstuvwxyz0123456789+/-');
7424: Const('b64_UUEncode','String').String('!"#$%&"()*+,./0123456789:;<>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[\]^_');
7425: Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNPQRSTUVWXYZZabedefghijklnopqrstuvwxyz');
7426: Const('CCHARSET','Stringb64_XXEncode');
7427: Const('CHEXSET','String'0123456789ABCDEF
7428: Const('HEXDIGITS','String'0123456789ABCDEF
7429: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7430: Const('DIGISET','String'0123456789
7431: Const('LETTERSET','String'ABCDEFIGHIJKLMNOPQRSTUVWXYZ'
7432: Const('DIGISET2','TCharset').SetSet('0123456789'
7433: Const('LETTERSET2','TCharset').SetSet('ABCDEFIGHIJKLMNOPQRSTUVWXYZ')
7434: Const('HEXSET2','TCharset').SetSet('0123456789ABCDEF');
7435: Const('NUMBERSET','TCharset').SetSet('0123456789');
7436: Const('NUMBERS','String'0123456789);
7437: Const('LETTERS','String'ABCDEFIGHIJKLMNOPQRSTUVWXYZ');
7438: Const('NUMBLETTSET','String').SetString('0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ');
7439: Const('NUMBLETTSET','TCharset').SetSet('0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ');
7440: Function CharSetToStr( const C : CharSet ) : AnsiString
7441: Function StrToCharSet( const S : AnsiString ) : CharSet
7442: Function MIMEBase64Decode( const S : AnsiString ) : AnsiString
7443: Function MIMEBase64Encode( const S : AnsiString ) : AnsiString
7444: Function UUDecode( const S : AnsiString ) : AnsiString
7445: Function XXDecode( const S : AnsiString ) : AnsiString
7446: Function BytesToHex( const P : array of Byte; const UpperCase : Boolean ) : AnsiString
7447: Function InterfaceToStrA( const I : IInterface ) : WideString
7448: Function InterfaceToStrW( const I : IInterface ) : WideString
7449: Function InterfaceToStr( const I : IInterface ) : String
7450: Function ObjectClassName( const O : TObject ) : String
7451: Function ClassClassName( const C : TClass ) : String
7452: Function ObjectToStr( const O : TObject ) : String
7453: Function ObjectToString( const O : TObject ) : String
7454: Function CharSetToStr( const C : CharSet ) : AnsiString
7455: Function StrToCharSet( const S : AnsiString ) : CharSet
7456: Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
    AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7457: Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
    AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7458: Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
    : Boolean; const Slots : LongWord) : LongWord
7459: Function HashInteger( const I : Integer; const Slots : LongWord ) : LongWord
7460: Function HashLongWord( const I : LongWord; const Slots : LongWord ) : LongWord
7461: Const('Bytes1KB','LongInt'( 1024));
7462: SIRegister_IInterface(CL);
7463: Procedure SelfTestCFundamentUtils
7464:
7465: Function CreateSchedule : IJclSchedule
7466: Function NullStamp : TTimeStamp
7467: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Int64
7468: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Boolean
7469: Function IsNullTimeStamp( const Stamp : TTimeStamp ) : Boolean
7470:
7471: procedure SIRegister_uwinplot(CL: TPPSPascalCompiler);
7472: begin
7473:   AddTypeS('TFunc', 'function(X : Float) : Float;');
7474:   Function InitGraphics( Width, Height : Integer ) : Boolean
7475:   Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean )
7476:   Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float )
7477:   Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float )
7478:   Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float )
7479:   Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float )
7480:   Procedure SetGraphTitle( Title : String )
7481:   Procedure SetOxTitle( Title : String )
7482:   Procedure SetOyTitle( Title : String )
7483:   Function GetGraphTitle : String
7484:   Function GetOxTitle : String
7485:   Function GetOyTitle : String
7486:   Procedure PlotOxAxis( Canvas : TCanvas )
7487:   Procedure PlotOyAxis( Canvas : TCanvas )
7488:   Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid )
7489:   Procedure WriteGraphTitle( Canvas : TCanvas )
7490:   Function SetMaxCurv( NCurv : Byte ) : Boolean
7491:   Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor )
7492:   Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor )
7493:   Procedure SetCurvLegend( CurvIndex : Integer; Legend : String )
7494:   Procedure SetCurvStep( CurvIndex, Step : Integer )
7495:   Function GetMaxCurv : Byte
7496:   Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor )
7497:   Procedure GetLineParam(CurvIndex:Intger;var Style:TPenStyle;var Width:Intger; var Color:TColor);
7498:   Function GetCurvLegend( CurvIndex : Integer ) : String
7499:   Function GetCurvStep( CurvIndex : Integer ) : Integer
7500:   Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer )
7501:   Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer )
7502:   Procedure PlotCurveWithErrorBars(Canvas: TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7503:   Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7504:   Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean )

```

```

7505: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7506: Function Xpixel( X : Float ) : Integer
7507: Function Ypixel( Y : Float ) : Integer
7508: Function Xuser( X : Integer ) : Float
7509: Function Yuser( Y : Integer ) : Float
7510: end;
7511:
7512: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7513: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7514: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7515: Procedure FFT_Integer_Cleanup
7516: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7517: //unit uPSI_JclStreams;
7518: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin ) : Int64
7519: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer ) : Int64
7520: Function CompareStreams( A, B : TStream; BufferSize : Integer ) : Boolean
7521: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer ) : Boolean
7522:
7523: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7524: begin
7525:   FindClass('TOBJECT','EInvalidDest'
7526:   FindClass('TOBJECT','EFCantMove'
7527:   Procedure fmxCopyFile( const FileName, DestName : string )
7528:   Procedure fmxMoveFile( const FileName, DestName : string )
7529:   Function fmxGetFileSize( const FileName : string ) : LongInt
7530:   Function fmxFileDateTime( const FileName : string ) : TDateTime
7531:   Function fmxHasAttr( const FileName : string; Attr : Word ) : Boolean
7532:   Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ):THandle;
7533: end;
7534:
7535: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7536: begin
7537:   SIRegister_IFindFileIterator(CL);
7538:   Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7539: end;
7540:
7541: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7542: begin
7543:   Function SkipWhite( cp : PChar ) : PChar
7544:   Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string ) : PChar
7545:   Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string ) : PChar
7546:   Function ReadIdent( cp : PChar; var ident : string ) : PChar
7547: end;
7548:
7549: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7550: begin
7551:   SIRegister_TStringHashMapTraits(CL);
7552:   Function CaseSensitiveTraits : TStringHashMapTraits
7553:   Function CaseInsensitiveTraits : TStringHashMapTraits
7554:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7555:   +'e; Right : PHashNode; end
7556:   //PHashArray', '^THashArray // will not work
7557:   SIRegister_TStringHashMap(CL);
7558:   THashValue', 'Cardinal
7559:   Function StrHash( const s : string ) : THashValue
7560:   Function TextHash( const s : string ) : THashValue
7561:   Function DataHash( var AValue, ASize : Cardinal ) : THashValue
7562:   Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer ) : Boolean
7563:   Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer ) : Boolean
7564:   Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer ) : Boolean
7565:   SIRegister_TCaseSensitiveTraits(CL);
7566:   SIRegister_TCaseInsensitiveTraits(CL);
7567:
7568:
7569: //*****unit uPSI_umath;
7570: Function uExp( X : Float ) : Float
7571: Function uExp2( X : Float ) : Float
7572: Function uExp10( X : Float ) : Float
7573: Function uLog( X : Float ) : Float
7574: Function uLog2( X : Float ) : Float
7575: Function uLog10( X : Float ) : Float
7576: Function uLogA( X, A : Float ) : Float
7577: Function uIntPower( X : Float; N : Integer): Float
7578: Function uPower( X, Y : Float ) : Float
7579: Function SgnGamma( X : Float ) : Integer
7580: Function Stirling( X : Float ) : Float
7581: Function StirLog( X : Float ) : Float
7582: Function Gamma( X : Float ) : Float
7583: Function LnGamma( X : Float ) : Float
7584: Function DiGamma( X : Float ) : Float
7585: Function TriGamma( X : Float ) : Float
7586: Function IGamma( X : Float ) : Float
7587: Function JGamma( X : Float ) : Float
7588: Function InvGamma( X : Float ) : Float
7589: Function Erf( X : Float ) : Float
7590: Function Erfc( X : Float ) : Float
7591: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7592: { Correlation coefficient between samples X and Y }
7593: function DBeta(A, B, X : Float) : Float;

```

```

7594: { Density of Beta distribution with parameters A and B }
7595: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7596: Function Beta(X, Y : Float) : Float
7597: Function Binomial( N, K : Integer) : Float
7598: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7599: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7600: Procedure LU_Decompo( A : TMatrix; Lb, Ub : Integer)
7601: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7602: Function DNorm( X : Float) : Float
7603:
7604: function DGamma(A, B, X : Float) : Float;
7605: { Density of Gamma distribution with parameters A and B }
7606: function DKhi2(Nu : Integer; X : Float) : Float;
7607: { Density of Khi-2 distribution with Nu d.o.f. }
7608: function DStudent(Nu : Integer; X : Float) : Float;
7609: { Density of Student distribution with Nu d.o.f. }
7610: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7611: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7612: function IBeta(A, B, X : Float) : Float;
7613: { Incomplete Beta function}
7614: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7615:
7616: procedure SIRegister_unlfit(CL: TPSPPascalCompiler);
7617: begin
7618:   Procedure SetOptAlgo( Algo : TOptAlgo)
7619:   procedure SetOptAlgo(Algo : TOptAlgo);
7620:   {
7621:     Sets the optimization algorithm according to Algo, which must be
7622:     NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ
7623:   }
7624:   Function GetOptAlgo : TOptAlgo
7625:   Procedure SetMaxParam( N : Byte)
7626:   Function GetMaxParam : Byte
7627:   Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7628:   Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7629:   Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7630:   Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDervProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
7631:     Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7632:   Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDervProc; X, Y, S : TVector; Lb, Ub:Integer;
7633:     MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7634:   Procedure SetMCFile( FileName : String)
7635:   Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7636:   Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
7637:     LastPar:Integer;V:TMatrix);
7638:   (*-----*)
7639:   begin
7640:     Procedure SaveSimplex( FileName : string)
7641:     Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7642:   end;
7643:   (*-----*)
7644:   procedure SIRegister_uregtest(CL: TPSPPascalCompiler);
7645:   begin
7646:     Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7647:     Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7648:   end;
7649:
7650:   procedure SIRegister_ustrings(CL: TPSPPascalCompiler);
7651:   begin
7652:     Function LTrim( S : String) : String
7653:     Function RTrim( S : String) : String
7654:     Function uTrim( S : String) : String
7655:     Function StrChar( N : Byte; C : Char) : String
7656:     Function RFill( S : String; L : Byte) : String
7657:     Function LFill( S : String; L : Byte) : String
7658:     Function CFill( S : String; L : Byte) : String
7659:     Function Replace( S : String; C1, C2 : Char) : String
7660:     Function Extract( S : String; var Index : Byte; Delim : Char) : String
7661:     Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7662:     Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7663:     Function FloatStr( X : Float) : String
7664:     Function IntStr( N : LongInt) : String
7665:     Function uCompStr( Z : Complex) : String
7666:   end;
7667:
7668:   procedure SIRegister_uhyper(CL: TPSPPascalCompiler);
7669:   begin
7670:     Function uSinh( X : Float) : Float
7671:     Function uCosh( X : Float) : Float
7672:     Function uTanh( X : Float) : Float
7673:     Function uArcSinh( X : Float) : Float
7674:     Function uArcCosh( X : Float) : Float
7675:     Function ArcTanh( X : Float) : Float
7676:     Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7677:   end;
7678:
7679:   procedure SIRegister_urandom(CL: TPSPPascalCompiler);

```

```

7680: begin
7681: type RNG_Type =
7682:   (RNG_MWC, { Multiply-With-Carry }
7683:    RNG_MT, { Mersenne Twister }
7684:    RNG_UVAG); { Universal Virtual Array Generator }
7685: Procedure SetRNG( RNG : RNG_Type)
7686: Procedure InitGen( Seed : RNG_IntType)
7687: Procedure SRand( Seed : RNG_IntType)
7688: Function IRanGen : RNG_IntType
7689: Function IRanGen31 : RNG_IntType
7690: Function RanGen1 : Float
7691: Function RanGen2 : Float
7692: Function RanGen3 : Float
7693: Function RanGen53 : Float
7694: end;
7695;
7696: // Optimization by Simulated Annealing
7697: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7698: begin
7699:   Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7700:   Procedure SA_CreateLogFile( FileName : String)
7701:   Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7702: end;
7703;
7704: procedure SIRegister_urauvag(CL: TPSPascalCompiler);
7705: begin
7706:   Procedure InitUVAGbyString( KeyPhrase : string)
7707:   Procedure InitUVAG( Seed : RNG_IntType)
7708:   Function IRanUVAG : RNG_IntType
7709: end;
7710;
7711: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7712: begin
7713:   Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7714:   Procedure GA_CreateLogFile( LogFileName : String)
7715:   Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7716: end;
7717;
7718: TVector', 'array of Float
7719: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7720: begin
7721:   Procedure QSort( X : TVector; Lb, Ub : Integer)
7722:   Procedure DQSort( X : TVector; Lb, Ub : Integer)
7723: end;
7724;
7725: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7726: begin
7727:   Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7728:   Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7729: end;
7730;
7731: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7732: begin
7733:   FT_Result', 'Integer
7734:   //TDWordptr', '^DWord // will not work
7735:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7736:   d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7737:   r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7738:   ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeUp : Word; Rev4 : B'
7739:   yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7740:   te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7741:   ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7742:   erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7743:   Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7744:   te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7745:   yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7746:   Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7747:   nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7748:   ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 : '
7749:   ; Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIIsVCP : B'
7750:   yte; end
7751: end;
7752;
7753;
7754: //***** PaintFX*****
7755: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7756: begin
7757:   //with RegClass(CL, 'TComponent', 'TJvPaintFX') do
7758:   with AddClassN(FindClass('TComponent'), 'TJvPaintFX') do begin
7759:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7760:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7761:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7762:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7763:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7764:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7765:     Procedure Turn( Src, Dst : TBitmap)
7766:     Procedure TurnRight( Src, Dst : TBitmap)
7767:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7768:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)

```

```

7769: Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7770: Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7771: Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7772: Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7773: Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7774: Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7775: Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7776: Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7777: Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7778: Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7779: Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7780: Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7781: Procedure Emboss( var Bmp : TBitmap)
7782: Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7783: Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7784: Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7785: Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7786: Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7787: Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7788: Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7789: Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7790: Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7791: Procedure QuartoOpaque( Src, Dst : TBitmap)
7792: Procedure SemiOpaque( Src, Dst : TBitmap)
7793: Procedure ShadowDownLeft( const Dst : TBitmap)
7794: Procedure ShadowDownRight( const Dst : TBitmap)
7795: Procedure ShadowUpLeft( const Dst : TBitmap)
7796: Procedure ShadowUpRight( const Dst : TBitmap)
7797: Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7798: Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7799: Procedure FlipRight( const Dst : TBitmap)
7800: Procedure FlipDown( const Dst : TBitmap)
7801: Procedure Spotlight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7802: Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7803: Procedure MakeSeamlessClip( var Dst : TBitmap; Sean : Integer)
7804: Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7805: Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7806: Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7807: Procedure SmoothResize( var Src, Dst : TBitmap)
7808: Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7809: Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7810: Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7811: Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7812: Procedure GrayScale( const Dst : TBitmap)
7813: Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7814: Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7815: Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7816: Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7817: Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7818: Procedure Spray( const Dst : TBitmap; Amount : Integer)
7819: Procedure AntiAlias( const Dst : TBitmap)
7820: Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7821: Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7822: Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7823: Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7824: Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7825: Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7826: Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7827: Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7828: Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7829: Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7830: Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7831: Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7832: Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7833: Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7834: Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7835: Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7836: Procedure Invert( Src : TBitmap)
7837: Procedure MirrorRight( Src : TBitmap)
7838: Procedure MirrorDown( Src : TBitmap)
7839: end;
7840: end;
7841:
7842: (*-----*)
7843: procedure SJRegister_JvPaintFX(CL: TPSPPascalCompiler);
7844: begin
7845:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7846:             +'ye, lbtwist, lbrimble, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7847:             +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )'
7848:   SJRegister_TJvPaintFX(CL);
7849:   Function SplineFilter( Value : Single) : Single
7850:   Function BellFilter( Value : Single) : Single
7851:   Function TriangleFilter( Value : Single) : Single
7852:   Function BoxFilter( Value : Single) : Single
7853:   Function HermiteFilter( Value : Single) : Single
7854:   Function Lanczos3Filter( Value : Single) : Single
7855:   Function MitchellFilter( Value : Single) : Single
7856: end;
7857:

```

```

7858:
7859: (*-----*)
7860: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7861: begin
7862:   'TeeMsg_DefaultFunctionName', 'String 'TeeFunction
7863:   TeeMsg_DefaultSeriesName', 'String 'Series
7864:   TeeMsg_DefaultToolName', 'String 'ChartTool
7865:   ChartComponentPalette', 'String 'TeeChart
7866:   TeeMaxLegendColumns', 'LongInt'( 2 );
7867:   TeeDefaultLegendSymbolWidth', 'LongInt'( 20 );
7868:   TeeTitleFootDistance, LongInt( 5 );
7869:   SIRegister_TCustomChartWall(CL);
7870:   SIRegister_TChartWall(CL);
7871:   SIRegister_TChartLegendGradient(CL);
7872:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7873:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7874:   FindClass('TOBJECT'), 'LegendException
7875:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7876:     +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7877:   FindClass('TOBJECT'), 'TCustomChartLegend
7878:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7879:   TLegendSymbolPosition', '( spLeft, spRight )
7880:   TSymbolDrawEvent', 'Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect)';
7881:   TSymbolCalcHeight', 'Function : Integer
7882:   SIRegister_TLegendSymbol(CL);
7883:   SIRegister_TTeeCustomShapePosition(CL);
7884:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7885:   SIRegister_TLegendTitle(CL);
7886:   SIRegister_TLegendItem(CL);
7887:   SIRegister_TLegendItems(CL);
7888:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7889:   FindClass('TOBJECT'), 'TCustomChart
7890:   SIRegister_TCustomChartLegend(CL);
7891:   SIRegister_TChartLegend(CL);
7892:   SIRegister_TChartTitle(CL);
7893:   SIRegister_TChartFootTitle(CL);
7894:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7895:     +'eButton; Shift : TShiftState; X, Y : Integer)
7896:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7897:     +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7898:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series : '
7899:     +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7900:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7901:     +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7902:   TOnGetLegendPos', 'Procedure ( Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7903:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7904:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7905:     +'toMax : Boolean; Min : Double; Max : Double; end
7906:   TA11AxisSavedScales', 'array of TAxisSavedScales
7907:   SIRegister_TChartBackWall(CL);
7908:   SIRegister_TChartRightWall(CL);
7909:   SIRegister_TChartBottomWall(CL);
7910:   SIRegister_TChartLeftWall(CL);
7911:   SIRegister_TChartWalls(CL);
7912:   TChartAllowScrollEvent', 'Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean)';
7913:   SIRegister_TCustomChart(CL);
7914:   SIRegister_TChart(CL);
7915:   SIRegister_TTeeSeriesTypes(CL);
7916:   SIRegister_TTeeToolTypes(CL);
7917:   SIRegister_TTeeDragObject(CL);
7918:   SIRegister_TColorPalettes(CL);
7919:   Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
    AGalleryPage:PString;ANumGallerySeries:Integer;
7920:   Procedure RegisterTeeSeriesl( ASeriesClass : TChartSeriesClass; ADscription : PString);
7921:   Procedure RegisterTeeFunction(AFuncfClass:TTeeFunctionClass;ADescription,
    AGalleryPage:PString;ANumGallerySeries : Int;
7922:   Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADscription : PString)
7923:   Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
    ADscription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7924:   Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7925:   Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7926:   Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7927:   Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7928:   Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
    AFunctionClass : TTeeFunctionClass) : TChartSeries
7929:   Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7930:   Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7931:   Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7932:   Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent ) : TTeeCustomTool
7933:   Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7934:   Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7935:   Function GetNewSeriesName( AOwner : TComponent ) : TComponentName
7936:   Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7937:   Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7938:   Function GetGallerySeriesName( ASeries : TChartSeries) : String
7939:   Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
    R:TRect;ReferenceChart:TCustomChart);
7940:   SIRegister_TChartTheme(CL);
7941:   //TChartThemeClass', 'class of TChartTheme

```

```

7942: //TCanvasClass', 'class of TCanvas3D
7943: Function SeriesNameOrIndex( ASeries : TCustomChartSeries ) : String
7944: Function SeriesTitleOrName( ASeries : TCustomChartSeries ) : String
7945: Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean )
7946: Procedure ShowMessageUser( const S : String )
7947: Function HasNoMandatoryValues( ASeries : TChartSeries ) : Boolean
7948: Function HasLabels( ASeries : TChartSeries ) : Boolean
7949: Function HasColors( ASeries : TChartSeries ) : Boolean
7950: Function SeriesGuessContents( ASeries : TChartSeries ) : TeeFormatFlag
7951: Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer )
7952: end;
7953:
7954:
7955: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7956: begin
7957: // 'TeeFormBorderStyle',' bsNone );
7958: SIRegister_TMetafile(CL);
7959: 'TeeDefVerticalMargin','LongInt'( 4 );
7960: 'TeeDefHorizMargin','LongInt'( 3 );
7961: 'crTeeHand','LongInt'( TCursor( 2020 ) );
7962: 'TeeMsg_TeeHand','String 'crTeeHand
7963: 'TeeNormalPrintDetail','LongInt'( 0 );
7964: 'TeeHighPrintDetail','LongInt'( -100 );
7965: 'TeeDefault_PrintMargin','LongInt'( 15 );
7966: 'MaxDefaultColors','LongInt'( 19 );
7967: 'TeeTabDelimiter','Char #9';
7968: TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7969: +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7970: +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7971: +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7972: +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7973: +'ourMonths, dtSixMonths, dtOneYear, dtNone )'
7974: SIRegister_TCustomPanelNoCaption(CL);
7975: FindClass('TOBJECT'), 'TCustomTeePanel
7976: SIRegister_TZoomPanning(CL);
7977: SIRegister_TTeeEvent(CL);
7978: //SIRegister_TTeeEventListeners(CL);
7979: TTeeMouseEventKind', '( meDown, meUp, meMove )'
7980: SIRegister_TTeeMouseEvent(CL);
7981: SIRegister_TCustomTeePanel(CL);
7982: //TChartGradient', 'TTeeGradient
7983: //TChartGradientClass', 'class of TChartGradient
7984: TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )'
7985: SIRegister_TTeeZoomPen(CL);
7986: SIRegister_TTeeZoomBrush(CL);
7987: TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )'
7988: SIRegister_TTeeZoom(CL);
7989: FindClass('TOBJECT'), 'TCustomTeePanelExtended
7990: TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )'
7991: SIRegister_TBackImage(CL);
7992: SIRegister_TCustomTeePanelExtended(CL);
7993: //TChartBrushClass', 'class of TChartBrush
7994: SIRegister_TTeeCustomShapeBrushPen(CL);
7995: TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )'
7996: TTextFormat', '( ttfNormal, ttfHtml )'
7997: SIRegister_TTeeCustomShape(CL);
7998: SIRegister_TTeeShape(CL);
7999: SIRegister_TTeeExportData(CL);
8000: Function TeeStr( const Num : Integer ) : String
8001: Function DateTimeDefaultFormat( const AStep : Double ) : String
8002: Function TEEDaysInMonth( Year, Month : Word ) : Word
8003: Function FindDateTimeStep( const StepValue : Double ) : TDateTimeStep
8004: Function NextDateTimeStep( const AStep : Double ) : Double
8005: Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer ) : Boolean;
8006: Function PointInLine1( const P, FromPoint, ToPoint : TPoint ) : Boolean;
8007: Function PointInLine2( const P, FromPoint, ToPoint : TPoint; const TolerancePixels : Integer ) : Boolean;
8008: Function PointInLine3( const P : TPoint; const px, py, qx, qy : Integer ) : Boolean;
8009: Function PointInLineTolerance( const P : TPoint; const px, py, qx, qy, TolerancePixels : Integer ) : Boolean;
8010: Function PointInPolygon( const P : TPoint; const Poly : array of TPoint ) : Boolean;
8011: Function PointInTriangle( const P, P0, P1, P2 : TPoint ) : Boolean;
8012: Function PointInTriangle1( const P : TPoint; X0, Y0, X1, Y1 : Integer ) : Boolean;
8013: Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer ) : Boolean;
8014: Function PointInEllipse( const P : TPoint; const Rect : TRect ) : Boolean;
8015: Function PointInEllipse1( const P : TPoint; Left, Top, Right, Bottom : Integer ) : Boolean;
8016: Function DelphiToLocalFormat( const Format : String ) : String
8017: Function LocalToDelphiFormat( const Format : String ) : String
8018: Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
8019: Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8020: Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
  AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
8021: TTeeSortCompare', 'Function ( a, b : Integer ) : Integer
8022: TTeeSortSwap', 'Procedure ( a, b : Integer )
8023: Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8024: Function TeeGetUniqueName( AOwner : TComponent; const AStartTime : String ) : string
8025: Function TeeExtractField( St : String; Index : Integer ) : String;
8026: Function TeeExtractField1( St : String; Index : Integer; const Separator : String ) : String;
8027: Function TeeNumFields( St : String ) : Integer;
8028: Function TeeNumFields1( const St, Separator : String ) : Integer;
8029: Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap )

```

```

8030: Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
8031: // TColorArray', 'array of TColor
8032: Function GetDefaultColor( const Index : Integer) : TColor
8033: Procedure SetDefaultColorPalette;
8034: Procedure SetDefaultColorPalette( const Palette : array of TColor);
8035: 'TeeCheckBoxSize','LongInt'( 11);
8036: Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8037: Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
8038: Function TryStrToFloat( const S : String; var Value : Double) : Boolean
8039: Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
8040: Procedure TeeTranslateControl( AControl : TControl);
8041: Procedure TeeTranslateControl( AControl : TControl; const ExcludeChildren : array of TControl);
8042: Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
8043: //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8044: Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8045: //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8046: //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8047: Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8048: Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8049: Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8050: Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8051: Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8052: Procedure TeeSaveStringOption( const AKey, Value : String)
8053: Function TeeDefaultXMLEncoding : String
8054: Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8055: TeeWindowHandle', 'Integer
8056: Procedure TeeGoToURL( Handle : TeeWindowHandle; const URL : String)
8057: Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8058: Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8059: end;
8060:
8061:
8062: using mXBDEUtils
8063: ****
8064: Procedure SetAlias( aAlias, aDirectory : String)
8065: Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
Desired:Variant;Size:Byte);
8066: Function GetFileVersionNumber( const FileName : String) : TVersionNo
8067: Procedure SetBDE( aPath, aNode, aValue : String)
8068: function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8069: Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8070: Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8071: Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8072: Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8073: Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8074:
8075:
8076: procedure SIRegister_cDateTime(CL: TPPSPascalCompiler);
8077: begin
8078: AddClassN(FindClass('TOBJECT'),'EDateTime
8079: Function DatePart( const D : TDateTime) : Integer
8080: Function TimePart( const D : TDateTime) : Double
8081: Function Century( const D : TDateTime) : Word
8082: Function Year( const D : TDateTime) : Word
8083: Function Month( const D : TDateTime) : Word
8084: Function Day( const D : TDateTime) : Word
8085: Function Hour( const D : TDateTime) : Word
8086: Function Minute( const D : TDateTime) : Word
8087: Function Second( const D : TDateTime) : Word
8088: Function Millisecond( const D : TDateTime) : Word
8089: ('OneDay','Extended').setExtended( 1.0);
8090: ('OneHour','Extended').SetExtended( OneDay / 24);
8091: ('OneMinute','Extended').SetExtended( OneHour / 60);
8092: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8093: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8094: ('OneWeek','Extended').SetExtended( OneDay * 7);
8095: ('HoursPerDay','Extended').SetExtended( 24);
8096: ('MinutesPerHour','Extended').SetExtended( 60);
8097: ('SecondsPerMinute','Extended').SetExtended( 60);
8098: Procedure SetYear( var D : TDateTime; const Year : Word)
8099: Procedure SetMonth( var D : TDateTime; const Month : Word)
8100: Procedure SetDay( var D : TDateTime; const Day : Word)
8101: Procedure SetHour( var D : TDateTime; const Hour : Word)
8102: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8103: Procedure SetSecond( var D : TDateTime; const Second : Word)
8104: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8105: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8106: Function IsEqual( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8107: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8108: Function IsAM( const D : TDateTime) : Boolean
8109: Function IsPM( const D : TDateTime) : Boolean
8110: Function IsMidnight( const D : TDateTime) : Boolean
8111: Function IsNoon( const D : TDateTime) : Boolean
8112: Function IsSunday( const D : TDateTime) : Boolean
8113: Function IsMonday( const D : TDateTime) : Boolean
8114: Function IsTuesday( const D : TDateTime) : Boolean
8115: Function IsWednesday( const D : TDateTime) : Boolean
8116: Function IsThursday( const D : TDateTime) : Boolean
8117: Function IsFriday( const D : TDateTime) : Boolean

```

```

8118: Function IsSaturday( const D : TDateTime ) : Boolean
8119: Function IsWeekend( const D : TDateTime ) : Boolean
8120: Function Noon( const D : TDateTime ) : TDateTime
8121: Function Midnight( const D : TDateTime ) : TDateTime
8122: Function FirstDayOfMonth( const D : TDateTime ) : TDateTime
8123: Function LastDayOfMonth( const D : TDateTime ) : TDateTime
8124: Function NextWorkday( const D : TDateTime ) : TDateTime
8125: Function PreviousWorkday( const D : TDateTime ) : TDateTime
8126: Function FirstDayOfYear( const D : TDateTime ) : TDateTime
8127: Function LastDayOfYear( const D : TDateTime ) : TDateTime
8128: Function EasterSunday( const Year : Word ) : TDateTime
8129: Function GoodFriday( const Year : Word ) : TDateTime
8130: Function AddMilliseconds( const D : TDateTime; const N : Int64 ) : TDateTime
8131: Function AddSeconds( const D : TDateTime; const N : Int64 ) : TDateTime
8132: Function AddMinutes( const D : TDateTime; const N : Integer ) : TDateTime
8133: Function AddHours( const D : TDateTime; const N : Integer ) : TDateTime
8134: Function AddDays( const D : TDateTime; const N : Integer ) : TDateTime
8135: Function AddWeeks( const D : TDateTime; const N : Integer ) : TDateTime
8136: Function AddMonths( const D : TDateTime; const N : Integer ) : TDateTime
8137: Function AddYears( const D : TDateTime; const N : Integer ) : TDateTime
8138: Function DayOfYear( const Ye, Mo, Da : Word ) : Integer
8139: Function DayOfYear( const D : TDateTime ) : Integer
8140: Function DaysInMonth( const Ye, Mo : Word ) : Integer
8141: Function DaysInMonth( const D : TDateTime ) : Integer
8142: Function DaysInYear( const Ye : Word ) : Integer
8143: Function DaysInYearDate( const D : TDateTime ) : Integer
8144: Function WeekNumber( const D : TDateTime ) : Integer
8145: Function ISOFirstWeekOfYear( const Ye : Word ) : TDateTime
8146: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word )
8147: Function DiffMilliseconds( const D1, D2 : TDateTime ) : Int64
8148: Function DiffSeconds( const D1, D2 : TDateTime ) : Integer
8149: Function DiffMinutes( const D1, D2 : TDateTime ) : Integer
8150: Function DiffHours( const D1, D2 : TDateTime ) : Integer
8151: Function DiffDays( const D1, D2 : TDateTime ) : Integer
8152: Function DiffWeeks( const D1, D2 : TDateTime ) : Integer
8153: Function DiffMonths( const D1, D2 : TDateTime ) : Integer
8154: Function DiffYears( const D1, D2 : TDateTime ) : Integer
8155: Function GMTBias : Integer
8156: Function GMTTimeToLocalTime( const D : TDateTime ) : TDateTime
8157: Function LocalTimeToGMTTime( const D : TDateTime ) : TDateTime
8158: Function NowAsGMTTime : TDateTime
8159: Function DateTimeToISO8601String( const D : TDateTime ) : AnsiString
8160: Function ISO8601StringToTime( const D : AnsiString ) : TDateTime
8161: Function ISO8601StringAsDateTime( const D : AnsiString ) : TDateTime
8162: Function DateTimeToANSI( const D : TDateTime ) : Integer
8163: Function ANSIToDateTime( const Julian : Integer ) : TDateTime
8164: Function DateTimeToISOInteger( const D : TDateTime ) : Integer
8165: Function DateTimeToISOString( const D : TDateTime ) : AnsiString
8166: Function ISOIntegerToDate( const ISOInteger : Integer ) : TDateTime
8167: Function TwoDigitRadix2000YearToYear( const Y : Integer ) : Integer
8168: Function Date'imeAsElapsedTime( const D:TDateTime; const IncludeMilliseconds:Boolean ):AnsiString
8169: Function UnixTimeToDate( const UnixTime : LongWord ) : TDateTime
8170: Function DateTimeToUnixTime( const D : TDateTime ) : LongWord
8171: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer ) : AnsiString
8172: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer ) : UnicodeString
8173: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer ) : AnsiString
8174: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer ) : UnicodeString
8175: Function EnglishShortMonthStrA( const Month : Integer ) : AnsiString
8176: Function EnglishShortMonthStrU( const Month : Integer ) : UnicodeString
8177: Function EnglishLongMonthStrA( const Month : Integer ) : AnsiString
8178: Function EnglishLongMonthStrU( const Month : Integer ) : UnicodeString
8179: Function EnglishShortDayOfWeekA( const S : AnsiString ) : Integer
8180: Function EnglishShortDayOfWeekU( const S : UnicodeString ) : Integer
8181: Function EnglishLongDayOfWeekA( const S : AnsiString ) : Integer
8182: Function EnglishLongDayOfWeekU( const S : UnicodeString ) : Integer
8183: Function EnglishShortMonthA( const S : AnsiString ) : Integer
8184: Function EnglishShortMonthU( const S : UnicodeString ) : Integer
8185: Function EnglishLongMonthA( const S : AnsiString ) : Integer
8186: Function EnglishLongMonthU( const S : UnicodeString ) : Integer
8187: Function RFC850DayOfWeekA( const S : AnsiString ) : Integer
8188: Function RFC850DayOfWeekU( const S : UnicodeString ) : Integer
8189: Function RFC1123DayOfWeekA( const S : AnsiString ) : Integer
8190: Function RFC1123DayOfWeekU( const S : UnicodeString ) : Integer
8191: Function RFCMonthA( const S : AnsiString ) : Word
8192: Function RFCMonthU( const S : UnicodeString ) : Word
8193: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean ) : AnsiString
8194: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean ) : UnicodeString
8195: Function GMTDateTimeToRFC1123DateTimeA( const D : TDateTime; const IncludeDayOfWeek:Bool ):AnsiString;
8196: Function GMTDateTimeToRFC1123DateTimeU( const D : TDateTime; const IncludeDayOfWeek:Bool ):UnicodeString;
8197: Function DateTimeToRFCDateTimeA( const D : TDateTime ) : AnsiString
8198: Function DateTimeToRFCDateTimeU( const D : TDateTime ) : UnicodeString
8199: Function NowAsRFCDateTimeA : AnsiString
8200: Function NowAsRFCDateTimeU : UnicodeString
8201: Function RFCDateTimeToGMTDateTime( const S : AnsiString ) : TDateTime
8202: Function RFCDateTimeToDate( const S : AnsiString ) : TDateTime
8203: Function RFCTimeZoneToGMTBias( const Zone : AnsiString ) : Integer
8204: Function TimePeriodStr( const D : TDateTime ) : AnsiString
8205: Procedure SelfTest
8206: end;

```

```

8207: //*****C FileUtils
8208: Function PathHasDriveLetterA( const Path : AnsiString ) : Boolean
8209: Function PathHasDriveLetter( const Path : String ) : Boolean
8210: Function PathIsDriveLetterA( const Path : AnsiString ) : Boolean
8211: Function PathIsDriveLetter( const Path : String ) : Boolean
8212: Function PathIsDriveRootA( const Path : AnsiString ) : Boolean
8213: Function PathIsDriveRoot( const Path : String ) : Boolean
8214: Function PathIsRootA( const Path : AnsiString ) : Boolean
8215: Function PathIsRoot( const Path : String ) : Boolean
8216: Function PathIsUNCPathA( const Path : AnsiString ) : Boolean
8217: Function PathIsUNCPath( const Path : String ) : Boolean
8218: Function PathIsAbsoluteA( const Path : AnsiString ) : Boolean
8219: Function PathIsAbsolute( const Path : String ) : Boolean
8220: Function PathIsDirectoryA( const Path : AnsiString ) : Boolean
8221: Function PathIsDirectory( const Path : String ) : Boolean
8222: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8223: Function PathInclSuffix( const Path : String; const PathSep : Char ) : String
8224: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8225: Function PathExclSuffix( const Path : String; const PathSep : Char ) : String
8226: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char )
8227: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char )
8228: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char )
8229: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char )
8230: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8231: Function PathCanonical( const Path : String; const PathSep : Char ) : String
8232: Function PathExpandA( const Path:AnsiString;const basePath:AnsiString;const PathSep:Char):AnsiString
8233: Function PathExpand( const Path : String; const basePath : String; const PathSep : Char ) : String
8234: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8235: Function PathLeftElement( const Path : String; const PathSep : Char ) : String
8236: Procedure PathSplitLeftElementA( const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char );
8237: Procedure PathSplitLeftElement( const Path:String; var LeftElement,RightPath: String;const PathSep:Char );
8238: Procedure DecodeFilePathA( const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char );
8239: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char )
8240: Function FileNameValidA( const FileName : AnsiString ) : AnsiString
8241: Function FileNameValid( const FileName : String ) : String
8242: Function FilePathA( const FileName,Path:AnsiString;const basePath:AnsiStr;const PathSep:Char):AnsiString
8243: Function FilePath( const FileName, Path : String;const basePath: String;const PathSep : Char ) : String
8244: Function DirectoryExpandA( const Path:AnsiString;const basePath:AnsiString;const PathSep:Char):AnsiString
8245: Function DirectoryExpand( const Path : String; const basePath : String; const PathSep : Char ) : String
8246: Function UnixPathToWinPath( const Path : AnsiString ) : AnsiString
8247: Function WinPathToUnixPath( const Path : AnsiString ) : AnsiString
8248: Procedure CCopyFile( const FileName, DestName : String )
8249: Procedure CMoveFile( const FileName, DestName : String )
8250: Function CDeleteFiles( const FileMask : String ) : Boolean
8251: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8252: Procedure FileCloseEx( const FileHandle : TFileHandle )
8253: Function FileExistsA( const FileName : AnsiString ) : Boolean
8254: Function CFileExists( const FileName : String ) : Boolean
8255: Function CFileGetSize( const FileName : String ) : Int64
8256: Function FileGetDateTime( const FileName : String ) : TDateTime
8257: Function FileGetDateTime2( const FileName : String ) : TDateTime
8258: Function FileIsReadOnly( const FileName : String ) : Boolean
8259: Procedure FileDeleteEx( const FileName : String )
8260: Procedure FileRenameEx( const OldFileName, NewFileName : String )
8261: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode : TFileCreationMode; const FileOpenWait : PFileOpenWait ) : AnsiString
8262: Function DirectoryEntryExists( const Name : String ) : Boolean
8263: Function DirectoryEntrySize( const Name : String ) : Int64
8264: Function CDirectoryExists( const DirectoryName : String ) : Boolean
8265: Function DirectoryGetDateTime( const DirectoryName : String ) : TDateTime
8266: Procedure CDirectoryCreate( const DirectoryName : String )
8267: Function GetFirstFileNameMatching( const FileMask : String ) : String
8268: Function DirEntryGetAttr( const FileName : AnsiString ) : Integer
8269: Function DirEntryIsDirectory( const FileName : AnsiString ) : Boolean
8270: Function FileHasAttr( const FileName : String; const Attr : Word ) : Boolean
8271: AddTypes('TLogicalDriveType','( DriveRemovable, DriveFixed, DriveRemote,
8272: + DriveCDRom, DriveRamDisk, DriveTypeUnknown )')
8273: Function DriveIsValid( const Drive : Char ) : Boolean
8274: Function DriveGetType( const Path : AnsiString ) : TLogicalDriveType
8275: Function DriveFreeSpace( const Path : AnsiString ) : Int64
8276:
8277: procedure SIRegister_cTimers(CL: TPPSPascalCompiler);
8278: begin
8279: AddClassN(FindClass('TOBJECT'),'ETimers
8280: Const ('TickFrequency','LongInt'( 1000):Function GetTick : LongWord
8281: Function TickDelta( const D1, D2 : LongWord) : Integer
8282: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8283: AddTypes('THPTimer', 'Int64
8284: Procedure StartTimer( var Timer : THPTimer)
8285: Procedure StopTimer( var Timer : THPTimer)
8286: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8287: Procedure InitStoppedTimer( var Timer : THPTimer)
8288: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8289: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8290: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8291: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8292: Function GetHighPrecisionFrequency : Int64
8293: Function GetHighPrecisionTimerOverhead : Int64
8294: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)

```

```

8295: Procedure SelfTestCTimer
8296: end;
8297:
8298: procedure SIRегистер_cRandom(CL: TPSPPascalCompiler);
8299: begin
8300:   Function RandomSeed : LongWord
8301:   Procedure AddEntropy( const Value : LongWord)
8302:   Function RandomUniform : LongWord;
8303:   Function RandomUniforml( const N : Integer) : Integer;
8304:   Function RandomBoolean : Boolean
8305:   Function RandomByte : Byte
8306:   Function RandomByteNonZero : Byte
8307:   Function RandomWord : Word
8308:   Function RandomInt64 : Int64;
8309:   Function RandomInt64l( const N : Int64) : Int64;
8310:   Function RandomHex( const Digits : Integer) : String
8311:   Function RandomFloat : Extended
8312:   Function RandomAlphaStr( const Length : Integer) : AnsiString
8313:   Function RandomPseudoWord( const Length : Integer) : AnsiString
8314:   Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8315:   Function mwcRandomLongWord : LongWord
8316:   Function urnRandomLongWord : LongWord
8317:   Function moaRandomFloat : Extended
8318:   Function mwcRandomFloat : Extended
8319:   Function RandomNormalF : Extended
8320:   Procedure SelfTestCRandom
8321: end;
8322:
8323: procedure SIRегистер_SynEditMiscProcs(CL: TPSPPascalCompiler);
8324: begin
8325:   // PIntArray', '^TIntArray // will not work
8326:   Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8327:   TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8328:   Function synMax( x, y : integer ) : integer
8329:   Function synMin( x, y : integer ) : integer
8330:   Function synMinMax( x, mi, ma : integer ) : integer
8331:   Procedure synSwapInt( var l, r : integer )
8332:   Function synMaxPoint( const P1, P2 : TPoint ) : TPoint
8333:   Function synMinPoint( const P1, P2 : TPoint ) : TPoint
8334:   //Function synGetIntArray( Count : Cardinal; initialValue : integer ) : PIntArray
8335:   Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect )
8336:   Function synGetBestConvertTabsProc( TabWidth : integer ) : TConvertTabsProc
8337:   Function synConvertTabs( const Line : AnsiString; TabWidth : integer ) : AnsiString
8338:   Function synGetBestConvertTabsProcEx( TabWidth : integer ) : TConvertTabsProcEx
8339:   Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8340:   Function synGetExpandedLength( const aStr : string; aTabWidth : integer ) : integer
8341:   Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string ) : integer
8342:   Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8343:   Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8344:   Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8345:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8346:   +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8347:   ('C3_NONSPACING','LongInt'( 1 );
8348:   'C3_DIACRITIC','LongInt'( 2 );
8349:   'C3_VOWELMARK','LongInt'( 4 );
8350:   ('C3_SYMBOL','LongInt'( 8 );
8351:   ('C3_KATAKANA','LongWord( $0010);
8352:   ('C3_HIRAGANA','LongWord( $0020);
8353:   ('C3_HALFWIDTH','LongWord( $0040);
8354:   ('C3_FULLWIDTH','LongWord( $0080);
8355:   ('C3_IDEOGRAPH','LongWord( $0100);
8356:   ('C3_KASHIDA','LongWord( $0200);
8357:   ('C3_LEXICAL','LongWord( $0400);
8358:   ('C3_ALPHA','LongWord( $8000);
8359:   ('C3_NOTAPPLICABLE','LongInt'( 0 );
8360:   Function synStrScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8361:   Function synStrScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8362:   Function synIsStringType( Value : Word ) : TStringType
8363:   Function synGetEOL( Line : PChar ) : PChar
8364:   Function synEncodeString( s : string ) : string
8365:   Function synDecodeString( s : string ) : string
8366:   Procedure synFreeAndNil( var Obj: TObject )
8367:   Procedure synAssert( Expr : Boolean )
8368:   Function synLastDelimiter( const Delimiters, S : string ) : Integer
8369:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8370:   TReplaceFlags', 'set of TReplaceFlag )
8371:   Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8372:   Function synGetRValue( RGBValue : TColor ) : byte
8373:   Function synGetGValue( RGBValue : TColor ) : byte
8374:   Function synGetBValue( RGBValue : TColor ) : byte
8375:   Function synRGB( r, g, b : Byte ) : Cardinal
8376:   // THighlighterAttriProc', 'Function( Highlighter : TSynCustomHigh
8377:   // +'lighter; Attr:TSynHighlighterAttributes;UniqueAttrName:string;Params array of Pointer):Boolean;
8378:   //Function synEnumHighlighterAttrs( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
8379:   HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer ) : Boolean
8380:   Function synCalcFCS( const ABuf, ABufSize : Cardinal ) : Word
8381:   Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
     AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8381: end;

```

```

8382:
8383: Function GET_APPCOMMAND_LPARAM( lParam : LPARAM ) : WORD
8384: Function GET_DEVICE_LPARAM( lParam : LPARAM ) : WORD
8385: Function GET_KEYSTATE_LPARAM( lParam : LPARAM ) : WORD
8386:
8387: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8388: begin
8389:   Function STimeZoneBias : integer
8390:   Function TimeZone : string
8391:   Function Rfc822DateTime( t : TDateTime ) : string
8392:   Function CDateTime( t : TDateTime ) : string
8393:   Function SimpleDateTime( t : TDateTime ) : string
8394:   Function AnsiCDatetime( t : TDateTime ) : string
8395:   Function GetMonthNumber( Value : String ) : integer
8396:   Function GetTimeFromStr( Value : string ) : TDateTime
8397:   Function GetDateMDYFromStr( Value : string ) : TDateTime
8398:   Function DecodeRFCDateTime( Value : string ) : TDateTime
8399:   Function GetUTCTime : TDateTime
8400:   Function SetUTCTime( Newdt : TDateTime ) : Boolean
8401:   Function SGetTick : LongWord
8402:   Function STickDelta( TickOld, TickNew : LongWord ) : LongWord
8403:   Function CodeInt( Value : Word ) : Ansistring
8404:   Function DecodeInt( const Value : Ansistring; Index : Integer ) : Word
8405:   Function CodeLongInt( Value : LongInt ) : Ansistring
8406:   Function DecodeLongInt( const Value : Ansistring; Index : Integer ) : LongInt
8407:   Function DumpStr( const Buffer : Ansistring ) : string
8408:   Function DumpExStr( const Buffer : Ansistring ) : string
8409:   Procedure Dump( const Buffer : AnsiString; DumpFile : string )
8410:   Procedure DumpEx( const Buffer : AnsiString; DumpFile : string )
8411:   Function TrimSPLeft( const S : string ) : string
8412:   Function TrimSPRight( const S : string ) : string
8413:   Function TrimSP( const S : string ) : string
8414:   Function SeparateLeft( const Value, Delimiter : string ) : string
8415:   Function SeparateRight( const Value, Delimiter : string ) : string
8416:   Function SGetParameter( const Value, Parameter : string ) : string
8417:   Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings )
8418:   Procedure ParseParameters( Value : string; const Parameters : TStrings )
8419:   Function IndexByBegin( Value : string; const List : TStrings ) : integer
8420:   Function GetEmailAddr( const Value : string ) : string
8421:   Function GetEmailDesc( Value : string ) : string
8422:   Function CStrToHex( const Value : Ansistring ) : string
8423:   Function CIntToBin( Value : Integer; Digits : Byte ) : string
8424:   Function CBinToInt( const Value : string ) : Integer
8425:   Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string ):string
8426:   Function CReplaceString( Value, Search, Replace : AnsiString ) : AnsiString
8427:   Function CRPosEx( const Sub, Value : string; From : integer ) : Integer
8428:   Function CRPos( const Sub, Value : String ) : Integer
8429:   Function FetchBin( var Value : string; const Delimiter : string ) : string
8430:   Function CFetch( var Value : string; const Delimiter : string ) : string
8431:   Function FetchEx( var Value : string; const Delimiter, Quotation : string ) : string
8432:   Function IsBinaryString( const Value : AnsiString ) : Boolean
8433:   Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString ) : integer
8434:   Procedure StringsTrim( const value : TStrings )
8435:   Function PosStr( const SubStr, Value : String; From : integer ) : integer
8436:   Function IncPoint( const p : __pointer; Value : integer ) : __pointer
8437:   Function GetBetween( const PairBegin, PairEnd, Value : string ) : string
8438:   Function CCountOfChar( const Value : string; aChr : char ) : integer
8439:   Function UnquoteStr( const Value : string; Quote : Char ) : string
8440:   Function QuoteStr( const Value : string; Quote : Char ) : string
8441:   Procedure HeadersToList( const Value : TStrings )
8442:   Procedure ListToHeaders( const Value : TStrings )
8443:   Function SwapBytes( Value : integer ) : integer
8444:   Function ReadStrFromStream( const Stream : TStream; len : integer ) : AnsiString
8445:   Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString )
8446:   Function GetTempFile( const Dir, prefix : AnsiString ) : AnsiString
8447:   Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar ) : AnsiString
8448:   Function CXorString( Indata1, Indata2 : AnsiString ) : AnsiString
8449:   Function NormalizeHeader( Value : TStrings; var Index : Integer ) : string
8450: end;
8451:
8452: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8453: begin
8454:   ('CrcBufSize','LongInt'( 2048 );
8455:   Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
8456:   Function Adler32OfFile( Stream : TStream; CurCrc : LongInt ) : LongInt
8457:   Function Adler32OfFile( FileName : AnsiString ) : LongInt
8458:   Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
8459:   Function Crc16OfFile( Stream : TStream; CurCrc : Cardinal ) : Cardinal
8460:   Function Crc16OfFile( FileName : AnsiString ) : Cardinal
8461:   Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
8462:   Function Crc32OfFile( Stream : TStream; CurCrc : LongInt ) : LongInt
8463:   Function Crc32OfFile( FileName : AnsiString ) : LongInt
8464:   Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
8465:   Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
8466:   Function InternetSumOfFile( FileName : AnsiString ) : Cardinal
8467:   Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
8468:   Function Kermit16OfFile( Stream : TStream; CurCrc : Cardinal ) : Cardinal
8469:   Function Kermit16OfFile( FileName : AnsiString ) : Cardinal
8470: end;

```

```

8471:
8472: procedure SIRegister_CoObj(cl: TPSPascalCompiler);
8473: begin
8474:   function CreateOleObject(const ClassName: String): IDispatch;
8475:   function GetActiveOleObject(const ClassName: String): IDispatch;
8476:   function ProgIDToClassID(const ProgID: string): TGUID;
8477:   function ClassIDToProgID(const ClassID: TGUID): string;
8478:   function CreateClassID: string;
8479:   function CreateGUIDString: string;
8480:   function CreateGUIDID: string;
8481:   procedure OleError(ErrorCode: longint);
8482:   procedure OleCheck(Result: HResult);
8483: end;
8484:
8485: Function xCreateOleObject( const ClassName : string ) : Variant //or IDispatch
8486: Function xGetActiveOleObject( const ClassName : string ) : Variant
8487: //Function DllGetClassObject( const CLSID : TGUID; const IID : TIID; var Obj ) : HResult
8488: Function DllCanUnloadNow : HResult
8489: Function DllRegisterServer : HResult
8490: Function DllUnregisterServer : HResult
8491: Function VarFromInterface( Unknown : IUnknown ) : Variant
8492: Function VarToInterface( const V : Variant ) : IDispatch
8493: Function VarToAutoObject( const V : Variant ) : TAutoObject
8494: //Procedure
8495: DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer,Res:PVariant);
8496: //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo )
8497: Procedure OleError( ErrorCode : HResult )
8498: Procedure OleCheck( Result : HResult )
8499: Function StringToClassID( const S : string ) : TGUID
8500: Function ClassIDToString( const ClassID : TGUID ) : string
8501: Function xProgIDToClassID( const ProgID : string ) : TGUID
8502: Function xClassIDToProgID( const ClassID : TGUID ) : string
8503: Function xWideCompareStr( const S1, S2 : WideString ) : Integer
8504: Function xWideSameStr( const S1, S2 : WideString ) : Boolean
8505: Function xGUIDToString( const Classid : TGUID ) : string
8506: Function xStringToGUID( const S : string ) : TGUID
8507: Function xGetModuleName( Module : HMODULE ) : string
8508: Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer ) : Integer
8509: Function xUtf8Encode( const WS : WideString ) : UTF8String
8510: Function xUtf8Decode( const S : UTF8String ) : WideString
8511: Function xExcludeTrailingPathDelimiter( const S : string ) : string
8512: Function xIncludeTrailingPathDelimiter( const S : string ) : string
8513: Function XRTLHandleCOMException : HResult
8514: Procedure XRTLCheckArgument( Flag : Boolean )
8515: //Procedure XRTLCheckOutArgument( out Arg )
8516: Procedure XRTLInterfaceConnect( const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var Connection:Longint );
8517: Procedure XRTLInterfaceDisconnect( const Source: IUnknown; const IID:TIID;var Connection : Longint )
8518: Function XRTLRegisterActiveObject( const Unk:IUnknown;ClassID:TGUID;Flags:DWORD;var RegisterCookie:Int):HResult
8519: Function XRTLUUnRegisterActiveObject( var RegisterCookie : Integer ) : HResult
8520: //Function XRTLGetActiveObject( ClassID : TGUID; RIID : TIID; out Obj ) : HResult
8521: Procedure XRTLEnumActiveObjects( Strings : TStrings )
8522: function XRTLDefaultCategoryManager: IUnknown;
8523: function XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8524: // ICatRegister helper functions
8525: function XRTLCREATEComponentCategory(CatID: TGUID; CatDescription: WideString;
8526:                                         LocaleID: TGUID = LOCALE_USER_DEFAULT;
8527:                                         const CategoryManager: IUnknown = nil): HResult;
8528: function XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8529:                                         LocaleID: TGUID = LOCALE_USER_DEFAULT;
8530:                                         const CategoryManager: IUnknown = nil): HResult;
8531: function XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8532:                                         const CategoryManager: IUnknown = nil): HResult;
8533: function XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8534:                                         const CategoryManager: IUnknown = nil): HResult;
8535: // ICatInformation helper functions
8536: function XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8537:                                         LocaleID: TGUID = LOCALE_USER_DEFAULT;
8538:                                         const CategoryManager: IUnknown = nil): HResult;
8539: function XRTLGetCategoryList(Strings: TStrings; LocaleID: TGUID = LOCALE_USER_DEFAULT;
8540:                                         const CategoryManager: IUnknown = nil): HResult;
8541: function XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8542:                                         const CategoryManager: IUnknown = nil): HResult;
8543: function XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8544:                                         const CategoryManager: IUnknown = nil): HResult;
8545: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ');
8546:   const ADelete: Boolean = True): WideString;
8547: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8548: Function XRTLGetVariantAsString( const Value : Variant ) : string
8549: Function XRTLDateTimeToTimezoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone ) : TDateTime
8550: Function XRTLGetTimeZones : TXRTLTimeZones
8551: Function XFileTimeToDate( FileTime : TFileTime ) : TDateTime
8552: Function DateToFileTime( Date : TDateTime ) : TFileTime
8553: Function GMTNow : TDateTime
8554: Function GMTToLocalTime( GMT : TDateTime ) : TDateTime
8555: Function LocalTimeToGMT( LocalTime : TDateTime ) : TDateTime
8556: Procedure XRTLNotImplemented

```

```

8557: Procedure XRTLRaiseError( E : Exception);
8558: Procedure XRTLRaise( E : Exception)'');
8559: Procedure XRaise( E : Exception)'');
8560: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8561:
8562:
8563: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8564: begin
8565:   SIRegister_IXRTLValue(CL);
8566:   SIRegister_TXRTLValue(CL);
8567:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8568:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8569:   Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8570:   Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8571:   Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal;
8572:   Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal;
8573:   Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8574:   Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8575:   Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer;
8576:   Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer;
8577:   Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8578:   Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8579:   Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64;
8580:   Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64;
8581:   Function XRTLValue3( const AValue : Single) : IXRTLValue;
8582:   Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8583:   Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single;
8584:   Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single;
8585:   Function XRTLValue4( const AValue : Double) : IXRTLValue;
8586:   Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8587:   Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double;
8588:   Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double;
8589:   Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8590:   Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8591:   Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended;
8592:   Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended;
8593:   Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8594:   Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8595:   Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8596:   //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8597:   Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8598:   Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8599:   Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8600:   Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString;
8601:   Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString;
8602:   Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8603:   Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8604:   Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8605:   Function XRTLGetAsObjectDef( const IValue:IXRTLValue;const DefValue:TObject;const
     ADetachOwnership:Boolean):TObject;
8606:   //Function XRTLValue9( const AValue : _Pointer) : IXRTLValue;
8607:   //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : _Pointer) : _Pointer;
8608:   //Function XRTLGetAsPointer( const IValue : IXRTLValue) : _Pointer;
8609:   //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : _Pointer) : _Pointer;
8610:   Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8611:   Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8612:   Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant;
8613:   Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant;
8614:   Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8615:   Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8616:   Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency;
8617:   Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency;
8618:   Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8619:   Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8620:   Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp;
8621:   Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp;
8622:   Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8623:   Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8624:   Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass;
8625:   Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass;
8626:   Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8627:   Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8628:   Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID;
8629:   Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID;
8630:   Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8631:   Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8632:   Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean;
8633:   Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean;
8634: end;
8635:
8636: //*****unit uPSI_GR32;*****
8637:
8638: Function Color32( WinColor : TColor) : TColor32;
8639: Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8640: Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8641: Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32;
8642: Function WinColor( Color32 : TColor32) : TColor;
8643: Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32;
8644: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)

```

```

8645: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B, A : Byte)
8646: Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8647: Function RedComponent( Color32 : TColor32) : Integer
8648: Function GreenComponent( Color32 : TColor32) : Integer
8649: Function BlueComponent( Color32 : TColor32) : Integer
8650: Function AlphaComponent( Color32 : TColor32) : Integer
8651: Function Intensity( Color32 : TColor32) : Integer
8652: Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8653: Function HSLtoRGB( H, S, L : Single) : TColor32;
8654: Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8655: Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8656: Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8657: Function WinPalette( const P : TPalette32) : HPALETTE
8658: Function FloatPoint( X, Y : Single) : TFloatPoint;
8659: Function FloatPoint1( const P : TPoint) : TFloatPoint;
8660: Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8661: Function FixedPoint( X, Y : Integer) : TFixedPoint;
8662: Function FixedPoint1( X, Y : Single) : TFixedPoint;
8663: Function FixedPoint2( const P : TPoint) : TFixedPoint;
8664: Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8665: AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )')
8666: Function MakeRect( const L, T, R, B : Integer) : TRect;
8667: Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8668: Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8669: Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8670: Function FixedRect1( const ARect : TRect) : TRect;
8671: Function FixedRect2( const FR : TFloatRect) : TRect;
8672: Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8673: Function FloatRect1( const ARect : TRect) : TFloatRect;
8674: Function FloatRect2( const FXR : TRect) : TFloatRect;
8675: Function GIIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8676: Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8677: Function GUUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8678: Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8679: Function GEEqualRect( const R1, R2 : TRect) : Boolean;
8680: Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8681: Procedure GIInflateRect( var R : TRect; Dx, Dy : Integer);
8682: Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8683: Procedure GOFFsetRect( var R : TRect; Dx, Dy : Integer);
8684: Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8685: Function IsRectEmpty( const R : TRect) : Boolean;
8686: Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8687: Function GPTInRect( const R : TRect; const P : TPoint) : Boolean;
8688: Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8689: Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8690: Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8691: Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8692: Function EqualRectSized( const R1, R2 : TFloatRect) : Boolean;
8693: Function MessageBeep( uType : UINT) : BOOL
8694: Function ShowCursor( bShow : BOOL) : Integer
8695: Function SetCursorPos( X, Y : Integer) : BOOL
8696: Function SetCursor( hCursor : HICON) : HCURSOR
8697: Function GetCursorPos( var lpPoint : TPoint) : BOOL
8698: //Function ClipCursor( lpRect : PRect) : BOOL
8699: Function GetClipCursor( var lpRect : TRect) : BOOL
8700: Function GetCursor : HCURSOR
8701: Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8702: Function GetCaretBlinkTime : UINT
8703: Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8704: Function DestroyCaret : BOOL
8705: Function HideCaret( hWnd : HWND) : BOOL
8706: Function ShowCaret( hWnd : HWND) : BOOL
8707: Function SetCaretPos( X, Y : Integer) : BOOL
8708: Function GetCaretPos( var lpPoint : TPoint) : BOOL
8709: Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8710: Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8711: Function MapWindowPoints( hWndFrom, hWndTo : HWND; var lpPoints, cPoints : UINT) : Integer
8712: Function WindowFromPoint( Point : TPoint) : HWND
8713: Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8714:
8715:
8716: procedure SIRegister_GR32_Math(CL: TPPSPascalCompiler);
8717: begin
8718:   Function FixedFloor( A : TFixed) : Integer
8719:   Function FixedCeil( A : TFixed) : Integer
8720:   Function FixedMul( A, B : TFixed) : TFixed
8721:   Function FixedDiv( A, B : TFixed) : TFixed
8722:   Function OneOver( Value : TFixed) : TFixed
8723:   Function FixedRound( A : TFixed) : Integer
8724:   Function FixedSqr( Value : TFixed) : TFixed
8725:   Function FixedSqrtLP( Value : TFixed) : TFixed
8726:   Function FixedSqrtHP( Value : TFixed) : TFixed
8727:   Function FixedCombine( W, X, Y : TFixed) : TFixed
8728:   Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8729:   Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8730:   Function GRHypot( const X, Y : TFloat) : TFloat;
8731:   Function Hypot1( const X, Y : Integer) : Integer;
8732:   Function FastSqrt( const Value : TFloat) : TFloat
8733:   Function FastSqrtBab1( const Value : TFloat) : TFloat

```

```

8734: Function FastSqrtBab2( const Value : TFloat ) : TFloat
8735: Function FastInvSqrt( const Value : Single ) : Single;
8736: Function MulDiv( Multiplicand, Multiplier, Divisor : Integer ) : Integer
8737: Function GRIsPowerOf2( Value : Integer ) : Boolean
8738: Function PrevPowerOf2( Value : Integer ) : Integer
8739: Function NextPowerOf2( Value : Integer ) : Integer
8740: Function Average( A, B : Integer ) : Integer
8741: Function GRSign( Value : Integer ) : Integer
8742: Function FloatMod( x, y : Double ) : Double
8743: end;
8744:
8745: procedure SIRегистer_GR32_LowLevel(CL: TPSCompiler);
8746: begin
8747:   Function Clamp( const Value : Integer ) : Integer;
8748:   Procedure GRFillWord( var X, Count : Cardinal; Value : Longword )
8749:   Function StackAlloc( Size : Integer ) : Pointer
8750:   Procedure StackFree( P : Pointer )
8751:   Procedure Swap( var A, B : Pointer );
8752:   Procedure Swap1( var A, B : Integer );
8753:   Procedure Swap2( var A, B : TFixed );
8754:   Procedure Swap3( var A, B : TColor32 );
8755:   Procedure TestSwap( var A, B : Integer );
8756:   Procedure TestSwap1( var A, B : TFixed );
8757:   Function TestClip( var A, B : Integer; const Size : Integer ) : Boolean;
8758:   Function TestClip1( var A, B : Integer; const Start, Stop : Integer ) : Boolean;
8759:   Function GRConstrain( const Value, Lo, Hi : Integer ) : Integer;
8760:   Function Constrain1( const Value, Lo, Hi : Single ) : Single;
8761:   Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer ) : Integer
8762:   Function GRMin( const A, B, C : Integer ) : Integer;
8763:   Function GRMax( const A, B, C : Integer ) : Integer;
8764:   Function Clamp( Value, Max : Integer ) : Integer;
8765:   Function Clamp1( Value, Min, Max : Integer ) : Integer;
8766:   Function Wrap( Value, Max : Integer ) : Integer;
8767:   Function Wrap1( Value, Min, Max : Integer ) : Integer;
8768:   Function Wrap3( Value, Max : Single ) : Single;;
8769:   Function WrapPow2( Value, Max : Integer ) : Integer;
8770:   Function WrapPow21( Value, Min, Max : Integer ) : Integer;
8771:   Function Mirror( Value, Max : Integer ) : Integer;
8772:   Function Mirror1( Value, Min, Max : Integer ) : Integer;
8773:   Function MirrorPow2( Value, Max : Integer ) : Integer;
8774:   Function MirrorPow21( Value, Min, Max : Integer ) : Integer;
8775:   Function GetOptimalWrap( Max : Integer ) : TWrapProc;
8776:   Function GetOptimalWrap1( Min, Max : Integer ) : TWrapProcEx;
8777:   Function GetOptimalMirror( Max : Integer ) : TWrapProc;
8778:   Function GetOptimalMirror1( Min, Max : Integer ) : TWrapProcEx;
8779:   Function GetWrapProc( WrapMode : TWrapMode ) : TWrapProc;
8780:   Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer ) : TWrapProc;
8781:   Function GetWrapProcEx( WrapMode : TWrapMode ) : TWrapProcEx;
8782:   Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer ) : TWrapProcEx;
8783:   Function Div255( Value : Cardinal ) : Cardinal;
8784:   Function SAR_4( Value : Integer ) : Integer;
8785:   Function SAR_8( Value : Integer ) : Integer;
8786:   Function SAR_9( Value : Integer ) : Integer;
8787:   Function SAR_11( Value : Integer ) : Integer;
8788:   Function SAR_12( Value : Integer ) : Integer;
8789:   Function SAR_13( Value : Integer ) : Integer;
8790:   Function SAR_14( Value : Integer ) : Integer;
8791:   Function SAR_15( Value : Integer ) : Integer;
8792:   Function SAR_16( Value : Integer ) : Integer;
8793:   Function ColorSwap( WinColor : TColor ) : TColor32;
8794: end;
8795:
8796: procedure SIRегистer_GR32_Filters(CL: TPSCompiler);
8797: begin
8798:   AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )');
8799:   Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components );
8800:   Procedure CopyComponents1(Dst:TCustomBitmap32;DstX,
DstY:Int;Src:TCustomBitmap32;SrcRect:TRect;Components:TColor32Comp;
8801:   Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32 );
8802:   Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean );
8803:   Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32 );
8804:   Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components );
8805:   Procedure InvertRGB( Dst, Src : TCustomBitmap32 );
8806:   Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean );
8807:   Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32 );
8808:   Function CreateBitmask( Components : TColor32Components ) : TColor32;
8809:   Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
Bitmask : TColor32; LogicalOperator : TLogicalOperator );
8810:   Procedure
ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8811:   Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean );
8812: end;
8813:
8814:
8815: procedure SIRегистer_JclNTFS(CL: TPSCompiler);
8816: begin
8817:   AddClassN(FindClass('TOBJECT'), 'EJclNtfsError';
8818:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
8819:   Function NtfsGetCompression( const FileName : string; var State : Short ) : Boolean;

```

```

8820: Function NtfsGetCompression1( const FileName : string ) : TFileCompressionState;
8821: Function NtfsSetCompression( const FileName : string; const State : Short ) : Boolean
8822: Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState )
8823: Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8824: Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8825: Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8826: //AddTypeS('TNTfsAllocRanges', 'record Entries : Integer; Data : PFileAlloc'
8827: //+'+tedRangeBuffer; MoreData : Boolean; end
8828: Function NtfsSetSparse( const FileName : string ) : Boolean
8829: Function NtfsZeroDataByHandle( const Handle: THandle; const First, Last : Int64 ) : Boolean
8830: Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64 ) : Boolean
8831: //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
Ranges:TNTfsAllocRanges):Boolean;
8832: //Function NtfsGetAllocRangeEntry( const Ranges : TNTfsAllocRanges;
Index:Integer):TFileAllocatedRangeBuffer
8833: Function NtfsSparseStreamsSupported( const Volume : string ) : Boolean
8834: Function NtfsGetSparse( const FileName : string ) : Boolean
8835: Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD ) : Boolean
8836: Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword ) : Boolean
8837: //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8838: Function NtfsGetReparseTag( const Path : string; var Tag : DWORD ) : Boolean
8839: Function NtfsReparsePointsSupported( const Volume : string ) : Boolean
8840: Function NtfsFileHasReparsePoint( const Path : string ) : Boolean
8841: Function NtfsIsFolderMountPoint( const Path : string ) : Boolean
8842: Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char ) : Boolean
8843: Function NtfsMountVolume( const Volume : Char; const MountPoint : string ) : Boolean
8844: AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter ')
8845: Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8846: Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8847: Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8848: Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8849: Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped ) : Boolean
8850: Function NtfsCreateJunctionPoint( const Source, Destination : string ) : Boolean
8851: Function NtfsDeleteJunctionPoint( const Source : string ) : Boolean
8852: Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string ) : Boolean
8853: AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec '
+ 'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8854: AddTypeS('TStreamIds', 'set of TStreamId
8855: AddTypes('TInternalFindStreamData', 'record FileHandle : THandle; Context '
+ ': __Pointer; StreamIds : TStreamIds; end
8856: AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
+ 'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8857: Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8858: Function NtfsFindNextStream( var Data : TFindStreamData ) : Boolean
8859: Function NtfsFindStreamClose( var Data : TFindStreamData ) : Boolean
8860: Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string ) : Boolean
8861: AddTypeS('TNTfsHardLinkInfo', 'record LinkCount : Cardinal; FileInfo : Int64; end
8862: Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo ) : Boolean
8863: Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
List:TStrings):Bool;
8864: Function NtfsDeleteHardLinks( const FileName : string ) : Boolean
8865: Function JclAppInstances : TJclAppInstances;
8866: Function JclAppInstances1( const UniqueAppIdGuidStr : string ) : TJclAppInstances;
8867: Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND ) : TJclAppInstDataKind
8868: Procedure ReadMessageData( const Message : TMessage; var Data : __Pointer; var Size : Integer )
8869: Procedure ReadMessageString( const Message : TMessage; var S : string )
8870: Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings )
8871: 
8872: (*-----*)
8873: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8874: begin
8875:   FindClass('TOBJECT','EJclGraphicsError
8876:   TDynIntegerArrayArray', 'array of TDynIntegerArray
8877:   TDynPointArray', 'array of TPoint
8878:   TDynDynPointArrayArray', 'array of TDynPointArray
8879:   TPointF', 'record X : Single; Y : Single; end
8880:   TDynPointArrayF', 'array of TPointF
8881:   TDrawMode2', '( dmOpaque, dmBlend )
8882:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8883:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8884:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8885:   TMatrix3d', 'record array[0..2,0..2] of extended end
8886:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8887:   TScanLine', 'array of Integer
8888:   TScanLines', 'array of TScanLine
8889:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8890:   TGradientDirection', '( gdVertical, gdHorizontal )
8891:   TPolyFillMode', '( fmAlternate, fmWinding )
8892:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8893:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8894:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8895:   SIRegister_TJclDesktopCanvas(CL);
8896:   FindClass('TOBJECT','TJclRegion
8897:   SIRegister_TJclRegionInfo(CL);
8898:   SIRegister_TJclRegion(CL);
8899:   SIRegister_TJclThreadPersistent(CL);
8900:   SIRegister_TJclCustomMap(CL);
8901:   SIRegister_TJclBitmap32(CL);
8902: 
```

```

8906:  SIRегистер_TJclByteMap(CL);
8907:  SIRегистер_TJclTransformation(CL);
8908:  SIRегистер_TJclLinearTransformation(CL);
8909:  Procedure Stretch(NewWidth,
8910:    NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8911:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8912:  Function GetAntialiasedBitmap( const Bitmap : TBitmap ) : TBitmap;
8913:  Procedure BitmapToJpeg( const FileName : string );
8914:  Procedure JpegToBitmap( const FileName : string );
8915:  Function ExtractIconCount( const FileName : string ) : Integer;
8916:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer ) : HICON;
8917:  Function IconToBitmapJ( Icon : HICON ) : HBITMAP;
8918:  Procedure
8919:    BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode);
8920:    Procedure StretchTransfer(Dst:TJclBitmap32;
8921:      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode);
8922:    Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation );
8923:    Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect );
8924:    Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection : TGradientDirection) : Boolean;
8925:    Function CreateRegionFromBitmap(Bitmap : TBitmap; RegionColor:TColor;
8926:      RegionBitmapMode:TJclRegionBitmapMode) : HGRN;
8927:    Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND );
8928:    Procedure ScreenShot1( bm : TBitmap );
8929:    Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 );
8930:    Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 );
8931:    Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32 );
8932:    Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 );
8933:    Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 );
8934:    Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32 );
8935:    Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32 );
8936:    Procedure IntensityToAlpha( Dst, Src : TJclBitmap32 );
8937:    Procedure Invert( Dst, Src : TJclBitmap32 );
8938:    Procedure InvertRGB( Dst, Src : TJclBitmap32 );
8939:    Procedure ColorToGrayscale( Dst, Src : TJclBitmap32 );
8940:    Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8 );
8941:    Procedure SetGamma( Gamma : Single );
8942:  end;
8943:
8944:  (*-----*)
8945:  procedure SIRегистер_JclSynch(CL: TPSPPascalCompiler);
8946:  begin
8947:    Function LockedAdd( var Target : Integer; Value : Integer ) : Integer;
8948:    Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer ) : Integer;
8949:    Function LockedCompareExchange1( var Target : __Pointer; Exch, Comp : __Pointer ) : Pointer;
8950:    Function LockedDec( var Target : Integer ) : Integer;
8951:    Function LockedExchange( var Target : Integer; Value : Integer ) : Integer;
8952:    Function LockedExchangeAdd( var Target : Integer; Value : Integer ) : Integer;
8953:    Function LockedExchangeDec( var Target : Integer ) : Integer;
8954:    Function LockedExchangeInc( var Target : Integer ) : Integer;
8955:    Function LockedExchangeSub( var Target : Integer; Value : Integer ) : Integer;
8956:    Function LockedInc( var Target : Integer ) : Integer;
8957:    Function LockedSub( var Target : Integer; Value : Integer ) : Integer;
8958:    TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )';
8959:    SIRегистер_TJclDispatcherObject(CL);
8960:    Function WaitForMultipleObjects(const Objects:array of TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8961:    Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
8962:      TimeOut : Cardinal):Cardinal;
8963:    SIRегистер_TJclCriticalSection(CL);
8964:    SIRегистер_TJclCriticalSectionEx(CL);
8965:    SIRегистер_TJclEvent(CL);
8966:    SIRегистер_TJclWaitableTimer(CL);
8967:    SIRегистер_TJclSemaphore(CL);
8968:    SIRегистер_TJclMutex(CL);
8969:    POptexSharedInfo', '^POptexSharedInfo // will not work';
8970:    TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end';
8971:    SIRегистер_TJclOptex(CL);
8972:    TMrewPreferred', '( mpReaders, mpWriters, mpEqual )';
8973:    TMrewThreadInfo', 'record Threadid: Longword; RecursionCount: Integer; Reader : Boolean; end';
8974:    TMrewThreadInfoArray', 'array of TMrewThreadInfo';
8975:    SIRегистер_TJclMultiReadExclusiveWrite(CL);
8976:    PMetSectSharedInfo', '^PMetSectSharedInfo // will not work';
8977:    TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : ';
8978:      +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end';
8979:    PMeteredSection', '^PMeteredSection // will not work';
8980:    TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end';
8981:    SIRегистер_TJclMeteredSection(CL);
8982:    TEventInfo', 'record EventType : Longint; Signaled : LongBool; end';
8983:    TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end';
8984:    TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end';
8985:    TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end';
8986:    Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection ) : Boolean;
8987:    Function QueryEvent( Handle : THandle; var Info : TEventInfo ) : Boolean;
8988:    Function QueryMutex( Handle : THandle; var Info : TMutexInfo ) : Boolean;

```

```

8988: Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts ) : Boolean
8989: Function QueryTimer( Handle : THandle; var Info : TTimerInfo ) : Boolean
8990: FindClass('TOBJECT'), 'EJclWin32HandleObjectError
8991: FindClass('TOBJECT'), 'EJclDispatcherObjectError
8992: FindClass('TOBJECT'), 'EJclCriticalSectionError
8993: FindClass('TOBJECT'), 'EJclEventError
8994: FindClass('TOBJECT'), 'EJclWaitableTimerError
8995: FindClass('TOBJECT'), 'EJclSemaphoreError
8996: FindClass('TOBJECT'), 'EJclMutexError
8997: FindClass('TOBJECT'), 'EJclMeteredSectionError
8998: end;
8999:
9000:
9001: //*****unit uPSI_mORMotReport;
9002: Procedure SetCurrentPrinterAsDefault
9003: Function CurrentPrinterName : string
9004: Function mCurrentPrinterPaperSize : string
9005: Procedure UseDefaultPrinter
9006:
9007: procedure SIRegisterTSTREAM(C1: TPSCompiler);
9008: begin
9009:   with FindClass('TOBJECT'), 'TStream' do begin
9010:     IsAbstract := True;
9011:     //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
9012:     // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
9013:     function Read(Buffer:String;Count:LongInt):LongInt
9014:     function Write(Buffer:String;Count:LongInt):LongInt
9015:     function ReadString(Buffer:String;Count:LongInt):LongInt      //FileStream
9016:     function WriteString(Buffer:String;Count:LongInt):LongInt
9017:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
9018:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
9019:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt';
9020:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt';
9021:
9022:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
9023:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
9024:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
9025:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
9026:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
9027:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
9028:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
9029:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
9030:
9031:     function Seek(Offset:LongInt;Origin:Word):LongInt
9032:     procedure ReadBuffer(Buffer:String;Count:LongInt)
9033:     procedure WriteBuffer(Buffer:String;Count:LongInt)
9034:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt');
9035:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt');
9036:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt');
9037:     procedure WriteBufferFloat(Buffer:Double;Count:LongInt');
9038:
9039:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
9040:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9041:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9042:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9043:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9044:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9045:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9046:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9047:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9048:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9049:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9050:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9051:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9052:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9053:
9054:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
9055:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
9056:     procedure ReadBufferO(Buffer: TObject;Count:LongInt');
9057:     procedure WriteBufferO(Buffer: TObject;Count:LongInt');
9058:     //READBUFFERAC
9059:     function InstanceSize: Longint
9060:     Procedure FixupResourceHeader( FixupInfo : Integer )
9061:     Procedure ReadResHeader
9062:
9063: {$IFDEF DELPHI4UP}
9064:     function CopyFrom(Source:TStream;Count:Int64):LongInt
9065: {$ELSE}
9066:     function CopyFrom(Source:TStream;Count:Integer):LongInt
9067: {$ENDIF}
9068:     RegisterProperty('Position', 'LongInt', iptrw);
9069:     RegisterProperty('Size', 'LongInt', iptrw);
9070:   end;
9071: end;
9072:
9073:
9074: { ****
9075: Unit DMATH - Interface for DMATH.DLL
9076: **** }

```

```

9077: // see more docs/dmath_manual.pdf
9078:
9079: Function InitEval : Integer
9080: Procedure SetVariable( VarName : Char; Value : Float)
9081: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9082: Function Eval( ExpressionString : String) : Float
9083:
9084: unit dmath; //types are in built, others are external in DLL
9085: interface
9086: {$IFDEF DELPHI}
9087: uses
9088:   StdCtrls, Graphics;
9089: {$ENDIF}
9090: { -----
9091:   Types and constants
9092: ----- }
9093: {$i types.inc}
9094: { -----
9095:   Error handling
9096: ----- }
9097: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9098: { Sets the error code }
9099: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9100: { Sets error code and default function value }
9101: function MathErr : Integer; external 'dmath';
9102: { Returns the error code }
9103: { -----
9104:   Dynamic arrays
9105: ----- }
9106: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9107: { Sets the auto-initialization of arrays }
9108: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9109: { Creates floating point vector V[0..Ub] }
9110: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9111: { Creates integer vector V[0..Ub] }
9112: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9113: { Creates complex vector V[0..Ub] }
9114: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9115: { Creates boolean vector V[0..Ub] }
9116: procedure DimStrVector(var V : TStringVector; Ub : Integer); external 'dmath';
9117: { Creates string vector V[0..Ub] }
9118: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9119: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9120: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9121: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9122: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9123: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9124: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9125: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9126: procedure DimStrMatrix(var A : TStringMatrix; Ub1, Ub2 : Integer); external 'dmath';
9127: { Creates string matrix A[0..Ub1, 0..Ub2] }
9128: { -----
9129:   Minimum, maximum, sign and exchange
9130: ----- }
9131: function FMin(X, Y : Float) : Float; external 'dmath';
9132: { Minimum of 2 reals }
9133: function FMax(X, Y : Float) : Float; external 'dmath';
9134: { Maximum of 2 reals }
9135: function IMin(X, Y : Integer) : Integer; external 'dmath';
9136: { Minimum of 2 integers }
9137: function IMax(X, Y : Integer) : Integer; external 'dmath';
9138: { Maximum of 2 integers }
9139: function Sgn(X : Float) : Integer; external 'dmath';
9140: { Sign (returns 1 if X = 0) }
9141: function Sgn0(X : Float) : Integer; external 'dmath';
9142: { Sign (returns 0 if X = 0) }
9143: function DSgn(A, B : Float) : Float; external 'dmath';
9144: { Sgn(B) * |A| }
9145: procedure FSwap(var X, Y : Float); external 'dmath';
9146: { Exchange 2 reals }
9147: procedure ISwap(var X, Y : Integer); external 'dmath';
9148: { Exchange 2 integers }
9149: { -----
9150:   Rounding functions
9151: ----- }
9152: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9153: { Rounds X to N decimal places }
9154: function Ceil(X : Float) : Integer; external 'dmath';
9155: { Ceiling function }
9156: function Floor(X : Float) : Integer; external 'dmath';
9157: { Floor function }
9158: { -----
9159:   Logarithms, exponentials and power
9160: ----- }
9161: function Expo(X : Float) : Float; external 'dmath';
9162: { Exponential }
9163: function Exp2(X : Float) : Float; external 'dmath';
9164: { 2^X }
9165: function Exp10(X : Float) : Float; external 'dmath';

```

```

9166: {  $10^X$  }
9167: function Log(X : Float) : Float; external 'dmath';
9168: { Natural log }
9169: function Log2(X : Float) : Float; external 'dmath';
9170: { Log, base 2 }
9171: function Log10(X : Float) : Float; external 'dmath';
9172: { Decimal log }
9173: function LogA(X, A : Float) : Float; external 'dmath';
9174: { Log, base A }
9175: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9176: {  $X^N$  }
9177: function Power(X, Y : Float) : Float; external 'dmath';
9178: {  $X^Y, X \geq 0$  }
9179: { -----
9180:   Trigonometric functions
9181: ----- }
9182: function Pythag(X, Y : Float) : Float; external 'dmath';
9183: { Sqrt( $X^2 + Y^2$ ) }
9184: function FixAngle(Theta : Float) : Float; external 'dmath';
9185: { Set Theta in -Pi..Pi }
9186: function Tan(X : Float) : Float; external 'dmath';
9187: { Tangent }
9188: function ArcSin(X : Float) : Float; external 'dmath';
9189: { Arc sinus }
9190: function ArcCos(X : Float) : Float; external 'dmath';
9191: { Arc cosinus }
9192: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9193: { Angle (Ox, OM) with M(X,Y) }
9194: { -----
9195:   Hyperbolic functions
9196: ----- }
9197: function Sinh(X : Float) : Float; external 'dmath';
9198: { Hyperbolic sine }
9199: function Cosh(X : Float) : Float; external 'dmath';
9200: { Hyperbolic cosine }
9201: function Tanh(X : Float) : Float; external 'dmath';
9202: { Hyperbolic tangent }
9203: function ArcSinh(X : Float) : Float; external 'dmath';
9204: { Inverse hyperbolic sine }
9205: function ArcCosh(X : Float) : Float; external 'dmath';
9206: { Inverse hyperbolic cosine }
9207: function ArcTanh(X : Float) : Float; external 'dmath';
9208: { Inverse hyperbolic tangent }
9209: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9210: { Sinh & Cosh }
9211: { -----
9212:   Gamma function and related functions
9213: ----- }
9214: function Fact(N : Integer) : Float; external 'dmath';
9215: { Factorial }
9216: function SgnGamma(X : Float) : Integer; external 'dmath';
9217: { Sign of Gamma function }
9218: function Gamma(X : Float) : Float; external 'dmath';
9219: { Gamma function }
9220: function LnGamma(X : Float) : Float; external 'dmath';
9221: { Logarithm of Gamma function }
9222: function Stirling(X : Float) : Float; external 'dmath';
9223: { Stirling's formula for the Gamma function }
9224: function StirLog(X : Float) : Float; external 'dmath';
9225: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9226: function DiGamma(X : Float) : Float; external 'dmath';
9227: { Digamma function }
9228: function TriGamma(X : Float) : Float; external 'dmath';
9229: { Trigamma function }
9230: function IGamma(A, X : Float) : Float; external 'dmath';
9231: { Incomplete Gamma function }
9232: function JGamma(A, X : Float) : Float; external 'dmath';
9233: { Complement of incomplete Gamma function }
9234: function InvGamma(A, P : Float) : Float; external 'dmath';
9235: { Inverse of incomplete Gamma function }
9236: function Erf(X : Float) : Float; external 'dmath';
9237: { Error function }
9238: function Erfc(X : Float) : Float; external 'dmath';
9239: { Complement of error function }
9240: { -----
9241:   Beta function and related functions
9242: ----- }
9243: function Beta(X, Y : Float) : Float; external 'dmath';
9244: { Beta function }
9245: function IBeta(A, B, X : Float) : Float; external 'dmath';
9246: { Incomplete Beta function }
9247: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9248: { Inverse of incomplete Beta function }
9249: { -----
9250:   Lambert's function
9251: ----- }
9252: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9253: { -----
9254:   Binomial distribution

```

```

9255: ----- }
9256: function Binomial(N, K : Integer) : Float; external 'dmath';
9257: { Binomial coefficient C(N,K) }
9258: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9259: { Probability of binomial distribution }
9260: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9261: { Cumulative probability for binomial distrib. }
9262: { -----
9263: Poisson distribution
9264: ----- }
9265: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9266: { Probability of Poisson distribution }
9267: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9268: { Cumulative probability for Poisson distrib. }
9269: { -----
9270: Exponential distribution
9271: ----- }
9272: function DExpo(A, X : Float) : Float; external 'dmath';
9273: { Density of exponential distribution with parameter A }
9274: function FExpo(A, X : Float) : Float; external 'dmath';
9275: { Cumulative probability function for exponential dist. with parameter A }
9276: { -----
9277: Standard normal distribution
9278: ----- }
9279: function DNorm(X : Float) : Float; external 'dmath';
9280: { Density of standard normal distribution }
9281: function FNorm(X : Float) : Float; external 'dmath';
9282: { Cumulative probability for standard normal distrib. }
9283: function PNorm(X : Float) : Float; external 'dmath';
9284: { Prob(|U| > X) for standard normal distrib. }
9285: function InvNorm(P : Float) : Float; external 'dmath';
9286: { Inverse of standard normal distribution }
9287: { -----
9288: Student's distribution
9289: ----- }
9290: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9291: { Density of Student distribution with Nu d.o.f. }
9292: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9293: { Cumulative probability for Student distrib. with Nu d.o.f. }
9294: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9295: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9296: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9297: { Inverse of Student's t-distribution function }
9298: { -----
9299: Khi-2 distribution
9300: ----- }
9301: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9302: { Density of Khi-2 distribution with Nu d.o.f. }
9303: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9304: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9305: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9306: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9307: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9308: { Inverse of Khi-2 distribution function }
9309: { -----
9310: Fisher-Snedecor distribution
9311: ----- }
9312: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9313: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9314: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9315: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9316: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9317: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9318: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9319: { Inverse of Snedecor's F-distribution function }
9320: { -----
9321: Beta distribution
9322: ----- }
9323: function DBeta(A, B, X : Float) : Float; external 'dmath';
9324: { Density of Beta distribution with parameters A and B }
9325: function FBeta(A, B, X : Float) : Float; external 'dmath';
9326: { Cumulative probability for Beta distrib. with param. A and B }
9327: { -----
9328: Gamma distribution
9329: ----- }
9330: function DGamma(A, B, X : Float) : Float; external 'dmath';
9331: { Density of Gamma distribution with parameters A and B }
9332: function FGamma(A, B, X : Float) : Float; external 'dmath';
9333: { Cumulative probability for Gamma distrib. with param. A and B }
9334: { -----
9335: Expression evaluation
9336: ----- }
9337: function InitEval : Integer; external 'dmath';
9338: { Initializes built-in functions and returns their number }
9339: function Eval(ExpressionString : String) : Float; external 'dmath';
9340: { Evaluates an expression at run-time }
9341: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9342: { Assigns a value to a variable }
9343: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';

```

```

9344: { Adds a function to the parser }
9345: { -----
9346:   Matrices and linear equations
9347:   -----
9348: procedure GaussJordan(A          : TMatrix;
9349:                         Lb, Ubl, Ub2 : Integer;
9350:                         var Det      : Float); external 'dmath';
9351: { Transforms a matrix according to the Gauss-Jordan method }
9352: procedure LinEq(A          : TMatrix;
9353:                     B          : TVector;
9354:                     Lb, Ub   : Integer;
9355:                     var Det : Float); external 'dmath';
9356: { Solves a linear system according to the Gauss-Jordan method }
9357: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9358: { Cholesky factorization of a positive definite symmetric matrix }
9359: procedure LU_Decom(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9360: { LU decomposition }
9361: procedure LU_Solve(A       : TMatrix;
9362:                       B       : TVector;
9363:                       Lb, Ub : Integer;
9364:                       X       : TVector); external 'dmath';
9365: { Solution of linear system from LU decomposition }
9366: procedure QR_Decom(A       : TMatrix;
9367:                         Lb, Ubl, Ub2 : Integer;
9368:                         R       : TMatrix); external 'dmath';
9369: { QR decomposition }
9370: procedure QR_Solve(Q, R     : TMatrix;
9371:                         B       : TVector;
9372:                         Lb, Ubl, Ub2 : Integer;
9373:                         X       : TVector); external 'dmath';
9374: { Solution of linear system from QR decomposition }
9375: procedure SV_Decom(A       : TMatrix;
9376:                         Lb, Ubl, Ub2 : Integer;
9377:                         S       : TVector;
9378:                         V       : TMatrix); external 'dmath';
9379: { Singular value decomposition }
9380: procedure SV_SetZero(S    : TVector;
9381:                         Lb, Ub : Integer;
9382:                         Tol   : Float); external 'dmath';
9383: { Set lowest singular values to zero }
9384: procedure SV_Solve(U     : TMatrix;
9385:                         S     : TVector;
9386:                         V     : TMatrix;
9387:                         B     : TVector;
9388:                         Lb, Ubl, Ub2 : Integer;
9389:                         X     : TVector); external 'dmath';
9390: { Solution of linear system from SVD }
9391: procedure SV_Approx(U    : TMatrix;
9392:                         S    : TVector;
9393:                         V    : TMatrix;
9394:                         Lb, Ubl, Ub2 : Integer;
9395:                         A    : TMatrix); external 'dmath';
9396: { Matrix approximation from SVD }
9397: procedure EigenVals(A   : TMatrix;
9398:                         Lb, Ub : Integer;
9399:                         Lambda : TCompVector); external 'dmath';
9400: { Eigenvalues of a general square matrix }
9401: procedure EigenVect(A  : TMatrix;
9402:                         Lb, Ub : Integer;
9403:                         Lambda : TCompVector;
9404:                         V   : TMatrix); external 'dmath';
9405: { Eigenvalues and eigenvectors of a general square matrix }
9406: procedure EigenSym(A  : TMatrix;
9407:                         Lb, Ub : Integer;
9408:                         Lambda : TVector;
9409:                         V   : TMatrix); external 'dmath';
9410: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9411: procedure Jacobi(A   : TMatrix;
9412:                         Lb, Ub, MaxIter : Integer;
9413:                         Tol   : Float;
9414:                         Lambda : TVector;
9415:                         V   : TMatrix); external 'dmath';
9416: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9417: { -----
9418:   Optimization
9419:   -----
9420: procedure MinBrack(Func      : TFunc;
9421:                         var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9422: { Brackets a minimum of a function }
9423: procedure GoldSearch(Func    : TFunc;
9424:                         A, B     : Float;
9425:                         MaxIter : Integer;
9426:                         Tol   : Float;
9427:                         var Xmin, Ymin : Float); external 'dmath';
9428: { Minimization of a function of one variable (golden search) }
9429: procedure LinMin(Func      : TFuncNVar;
9430:                         X, DeltaX : TVector;
9431:                         Lb, Ub   : Integer;
9432:                         var R     : Float;

```

```

9433:           MaxIter : Integer;
9434:           Tol : Float;
9435:           var F_min : Float); external 'dmath';
9436: { Minimization of a function of several variables along a line }
9437: procedure Newton(Func : TFuncNVar;
9438:           HessGrad : THessGrad;
9439:           X : TVector;
9440:           Lb, Ub : Integer;
9441:           MaxIter : Integer;
9442:           Tol : Float;
9443:           var F_min : Float;
9444:           G : TVector;
9445:           H_inv : TMatrix;
9446:           var Det : Float); external 'dmath';
9447: { Minimization of a function of several variables (Newton's method) }
9448: procedure SaveNewton(FileName : string); external 'dmath';
9449: { Save Newton iterations in a file }
9450: procedure Marquardt(Func : TFuncNVar;
9451:           HessGrad : THessGrad;
9452:           X : TVector;
9453:           Lb, Ub : Integer;
9454:           MaxIter : Integer;
9455:           Tol : Float;
9456:           var F_min : Float;
9457:           G : TVector;
9458:           H_inv : TMatrix;
9459:           var Det : Float); external 'dmath';
9460: { Minimization of a function of several variables (Marquardt's method) }
9461: procedure SaveMarquardt(FileName : string); external 'dmath';
9462: { Save Marquardt iterations in a file }
9463: procedure BFGS(Func : TFuncNVar;
9464:           Gradient : TGradient;
9465:           X : TVector;
9466:           Lb, Ub : Integer;
9467:           MaxIter : Integer;
9468:           Tol : Float;
9469:           var F_min : Float;
9470:           G : TVector;
9471:           H_inv : TMatrix); external 'dmath';
9472: { Minimization of a function of several variables (BFGS method) }
9473: procedure SaveBFGS(FileName : string); external 'dmath';
9474: { Save BFGS iterations in a file }
9475: procedure Simplex(Func : TFuncNVar;
9476:           X : TVector;
9477:           Lb, Ub : Integer;
9478:           MaxIter : Integer;
9479:           Tol : Float;
9480:           var F_min : Float); external 'dmath';
9481: { Minimization of a function of several variables (Simplex) }
9482: procedure SaveSimplex(FileName : string); external 'dmath';
9483: { Save Simplex iterations in a file }
9484: { -----
9485: Nonlinear equations
9486: ----- }
9487: procedure RootBrack(Func : TFunc;
9488:           var X, Y, FX, FY : Float); external 'dmath';
9489: { Brackets a root of function Func between X and Y }
9490: procedure Bisect(Func : TFunc;
9491:           var X, Y : Float;
9492:           MaxIter : Integer;
9493:           Tol : Float;
9494:           var F : Float); external 'dmath';
9495: { Bisection method }
9496: procedure Secant(Func : TFunc;
9497:           var X, Y : Float;
9498:           MaxIter : Integer;
9499:           Tol : Float;
9500:           var F : Float); external 'dmath';
9501: { Secant method }
9502: procedure NewtEq(Func, Deriv : TFunc;
9503:           var X : Float;
9504:           MaxIter : Integer;
9505:           Tol : Float;
9506:           var F : Float); external 'dmath';
9507: { Newton-Raphson method for a single nonlinear equation }
9508: procedure NewtEqs(Equations : TEquations;
9509:           Jacobian : TJacobian;
9510:           X, F : TVector;
9511:           Lb, Ub : Integer;
9512:           MaxIter : Integer;
9513:           Tol : Float); external 'dmath';
9514: { Newton-Raphson method for a system of nonlinear equations }
9515: procedure Broyden(Equations : TEquations;
9516:           X, F : TVector;
9517:           Lb, Ub : Integer;
9518:           MaxIter : Integer;
9519:           Tol : Float); external 'dmath';
9520: { Broyden's method for a system of nonlinear equations }
9521: { ----- }

```

```

9522:  Polynomials and rational fractions
9523:  -----
9524:  function Poly(X      : Float;
9525:           Coef   : TVector;
9526:           Deg    : Integer) : Float; external 'dmath';
9527: { Evaluates a polynomial }
9528:  function RFrac(X      : Float;
9529:           Coef   : TVector;
9530:           Deg1, Deg2 : Integer) : Float; external 'dmath';
9531: { Evaluates a rational fraction }
9532:  function RootPol1(A, B : Float;
9533:           var X : Float); external 'dmath';
9534: { Solves the linear equation A + B * X = 0 }
9535:  function RootPol2(Coef : TVector;
9536:           Z     : TCompVector) : Integer; external 'dmath';
9537: { Solves a quadratic equation }
9538:  function RootPol3(Coef : TVector;
9539:           Z     : TCompVector) : Integer; external 'dmath';
9540: { Solves a cubic equation }
9541:  function RootPol4(Coef : TVector;
9542:           Z     : TCompVector) : Integer; external 'dmath';
9543: { Solves a quartic equation }
9544:  function RootPol(Coef : TVector;
9545:           Deg   : Integer;
9546:           Z     : TCompVector) : Integer; external 'dmath';
9547: { Solves a polynomial equation }
9548:  function SetRealRoots(Deg : Integer;
9549:           Z     : TCompVector;
9550:           Tol  : Float) : Integer; external 'dmath';
9551: { Set the imaginary part of a root to zero }
9552:  procedure SortRoots(Deg : Integer;
9553:           Z     : TCompVector); external 'dmath';
9554: { Sorts the roots of a polynomial }
9555: { -----
9556: Numerical integration and differential equations
9557: -----
9558:  function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9559: { Integration by trapezoidal rule }
9560:  function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9561: { Integral from A to B }
9562:  function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9563: { Integral from 0 to B }
9564:  function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9565: { Convolution product at time T }
9566:  procedure ConvTrap(Func1, Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9567: { Convolution by trapezoidal rule }
9568:  procedure RKF45(F          : TDiffEqs;
9569:           Neqn        : Integer;
9570:           Y, Yp       : TVector;
9571:           var T        : Float;
9572:           Tout, RelErr, AbsErr : Float;
9573:           var Flag     : Integer); external 'dmath';
9574: { Integration of a system of differential equations }
9575: { -----
9576: Fast Fourier Transform
9577: -----
9578:  procedure FFT(NumSamples      : Integer;
9579:           InArray, OutArray : TCompVector); external 'dmath';
9580: { Fast Fourier Transform }
9581:  procedure IFFT(NumSamples      : Integer;
9582:           InArray, OutArray : TCompVector); external 'dmath';
9583: { Inverse Fast Fourier Transform }
9584:  procedure FFT_Integer(NumSamples : Integer;
9585:           RealIn, ImagIn : TIntVector;
9586:           OutArray   : TCompVector); external 'dmath';
9587: { Fast Fourier Transform for integer data }
9588:  procedure FFT_Integer_Cleanup; external 'dmath';
9589: { Clear memory after a call to FFT_Integer }
9590:  procedure CalcFrequency(NumSamples,
9591:           FrequencyIndex : Integer;
9592:           InArray       : TCompVector;
9593:           var FFT        : Complex); external 'dmath';
9594: { Direct computation of Fourier transform }
9595: { -----
9596: Random numbers
9597: -----
9598:  procedure SetRNG(RNG : RNG_Type); external 'dmath';
9599: { Select generator }
9600:  procedure InitGen(Seed : RNG_IntType); external 'dmath';
9601: { Initialize generator }
9602:  function IRanGen : RNG_IntType; external 'dmath';
9603: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9604:  function IRanGen31 : RNG_IntType; external 'dmath';
9605: { 31-bit random integer in [0 .. 2^31 - 1] }
9606:  function RanGen1 : Float; external 'dmath';
9607: { 32-bit random real in [0,1] }
9608:  function RanGen2 : Float; external 'dmath';
9609: { 32-bit random real in [0,1] }
9610:  function RanGen3 : Float; external 'dmath';

```

```

9611: { 32-bit random real in (0,1) }
9612: function RanGen53 : Float; external 'dmath';
9613: { 53-bit random real in [0,1) }
9614: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9615: { Initializes the 'Multiply with carry' random number generator }
9616: function IRanMWC : RNG_IntType; external 'dmath';
9617: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9618: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9619: { Initializes Mersenne Twister generator with a seed }
9620: procedure InitMTbyArray(InitKey : array of RNG_LongType;
9621:                           KeyLength : Word); external 'dmath';
9622: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9623: function IRanMT : RNG_IntType; external 'dmath';
9624: { Random integer from MT generator }
9625: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9626: { Initializes the UVAG generator with a string }
9627: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9628: { Initializes the UVAG generator with an integer }
9629: function IRanUVAG : RNG_IntType; external 'dmath';
9630: { Random integer from UVAG generator }
9631: function RanGaussStd : Float; external 'dmath';
9632: { Random number from standard normal distribution }
9633: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9634: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9635: procedure RanMult(M : TVector; L : TMatrix;
9636:                      Lb, Ub : Integer;
9637:                      X : TVector); external 'dmath';
9638: { Random vector from multinormal distribution (correlated) }
9639: procedure RanMultIndep(M, S : TVector;
9640:                           Lb, Ub : Integer;
9641:                           X : TVector); external 'dmath';
9642: { Random vector from multinormal distribution (uncorrelated) }
9643: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9644: { Initializes Metropolis-Hastings parameters }
9645: procedure GetMHParams(var NCycles, MaxSim, SavedSim:Integer); external 'dmath';
9646: { Returns Metropolis-Hastings parameters }
9647: procedure Hastings(Func : TFuncNVar;
9648:                       T : Float;
9649:                       X : TVector;
9650:                       V : TMatrix;
9651:                       Lb, Ub : Integer;
9652:                       Xmat : TMatrix;
9653:                       X_min : TVector;
9654:                       var F_min : Float); external 'dmath';
9655: { Simulation of a probability density function by Metropolis-Hastings }
9656: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9657: { Initializes Simulated Annealing parameters }
9658: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9659: { Initializes log file }
9660: procedure SimAnn(Func : TFuncNVar;
9661:                      X, Xmin, Xmax : TVector;
9662:                      Lb, Ub : Integer;
9663:                      var F_min : Float); external 'dmath';
9664: { Minimization of a function of several var. by simulated annealing }
9665: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9666: { Initializes Genetic Algorithm parameters }
9667: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9668: { Initializes log file }
9669: procedure GenAlg(Func : TFuncNVar;
9670:                      X, Xmin, Xmax : TVector;
9671:                      Lb, Ub : Integer;
9672:                      var F_min : Float); external 'dmath';
9673: { Minimization of a function of several var. by genetic algorithm }
9674: { -----
9675:   Statistics
9676:   ----- }
9677: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9678: { Mean of sample X }
9679: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9680: { Minimum of sample X }
9681: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9682: { Maximum of sample X }
9683: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9684: { Median of sample X }
9685: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9686: { Standard deviation estimated from sample X }
9687: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9688: { Standard deviation of population }
9689: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9690: { Correlation coefficient }
9691: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9692: { Skewness of sample X }
9693: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9694: { Kurtosis of sample X }
9695: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9696: { Quick sort (ascending order) }
9697: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9698: { Quick sort (descending order) }
9699: procedure Interval(x1, x2 : Float);

```

```

9700:           MinDiv, MaxDiv : Integer;
9701:           var Min, Max, Step : Float); external 'dmath';
9702: { Determines an interval for a set of values }
9703: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9704:           var XMin, XMax, XStep : Float); external 'dmath';
9705: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9706: procedure StudIndep(N1, N2 : Integer;
9707:           M1, M2, S1, S2 : Float;
9708:           var T : Float;
9709:           var DoF : Integer); external 'dmath';
9710: { Student t-test for independent samples }
9711: procedure StudPaired(X, Y : TVector;
9712:           Lb, Ub : Integer;
9713:           var T : Float;
9714:           var DoF : Integer); external 'dmath';
9715: { Student t-test for paired samples }
9716: procedure AnOVal(Ns : Integer;
9717:           N : TIntVector;
9718:           M, S : TVector;
9719:           var V_f, V_r, F : Float;
9720:           var DoF_f, DoF_r : Integer); external 'dmath';
9721: { One-way analysis of variance }
9722: procedure AnOva2(NA, NB, Nobs : Integer;
9723:           M, S : TMatrix;
9724:           V, F : TVector;
9725:           DoF : TIntVector); external 'dmath';
9726: { Two-way analysis of variance }
9727: procedure Snedecor(N1, N2 : Integer;
9728:           S1, S2 : Float;
9729:           var F : Float;
9730:           var DoF1, DoF2 : Integer); external 'dmath';
9731: { Snedecor's F-test (comparison of two variances) }
9732: procedure Bartlett(Ns : Integer;
9733:           N : TIntVector;
9734:           S : TVector;
9735:           var Khi2 : Float;
9736:           var DoF : Integer); external 'dmath';
9737: { Bartlett's test (comparison of several variances) }
9738: procedure Mann_Whitney(N1, N2 : Integer;
9739:           X1, X2 : TVector;
9740:           var U, Eps : Float); external 'dmath';
9741: { Mann-Whitney test }
9742: procedure Wilcoxon(X, Y : TVector;
9743:           Lb, Ub : Integer;
9744:           var Ndiff : Integer;
9745:           var T, Eps : Float); external 'dmath';
9746: { Wilcoxon test }
9747: procedure Kruskal_Wallis(Ns : Integer;
9748:           N : TIntVector;
9749:           X : TMatrix;
9750:           var H : Float;
9751:           var DoF : Integer); external 'dmath';
9752: { Kruskal-Wallis test }
9753: procedure Khi2_Conform(N_cls : Integer;
9754:           N_estim : Integer;
9755:           Obs : TIntVector;
9756:           Calc : TVector;
9757:           var Khi2 : Float;
9758:           var DoF : Integer); external 'dmath';
9759: { Khi-2 test for conformity }
9760: procedure Khi2_Indep(N_lin : Integer;
9761:           N_col : Integer;
9762:           Obs : TIntMatrix;
9763:           var Khi2 : Float;
9764:           var DoF : Integer); external 'dmath';
9765: { Khi-2 test for independence }
9766: procedure Woolf_Conform(N_cls : Integer;
9767:           N_estim : Integer;
9768:           Obs : TIntVector;
9769:           Calc : TVector;
9770:           var G : Float;
9771:           var DoF : Integer); external 'dmath';
9772: { Woolf's test for conformity }
9773: procedure Woolf_Indep(N_lin : Integer;
9774:           N_col : Integer;
9775:           Obs : TIntMatrix;
9776:           var G : Float;
9777:           var DoF : Integer); external 'dmath';
9778: { Woolf's test for independence }
9779: procedure DimStatClassVector(var C : TStatClassVector;
9780:           Ub : Integer); external 'dmath';
9781: { Allocates an array of statistical classes: C[0..Ub] }
9782: procedure Distrib(X : TVector;
9783:           Lb, Ub : Integer;
9784:           A, B, H : Float;
9785:           C : TStatClassVector); external 'dmath';
9786: { Distributes an array X[Lb..Ub] into statistical classes }
9787: { -----
9788:   Linear / polynomial regression

```

```

9789: ----- }
9790: procedure LinFit(X, Y : TVector;
9791:                      Lb, Ub : Integer;
9792:                      B : TVector;
9793:                      V : TMatrix); external 'dmath';
9794: { Linear regression :  $Y = B(0) + B(1) * X$  }
9795: procedure WLinFit(X, Y, S : TVector;
9796:                      Lb, Ub : Integer;
9797:                      B : TVector;
9798:                      V : TMatrix); external 'dmath';
9799: { Weighted linear regression :  $Y = B(0) + B(1) * X$  }
9800: procedure SVDLinFit(X, Y : TVector;
9801:                      Lb, Ub : Integer;
9802:                      SVDTol : Float;
9803:                      B : TVector;
9804:                      V : TMatrix); external 'dmath';
9805: { Unweighted linear regression by singular value decomposition }
9806: procedure WSVDLinFit(X, Y, S : TVector;
9807:                      Lb, Ub : Integer;
9808:                      SVDTol : Float;
9809:                      B : TVector;
9810:                      V : TMatrix); external 'dmath';
9811: { Weighted linear regression by singular value decomposition }
9812: procedure MulFit(X : TMatrix;
9813:                      Y : TVector;
9814:                      Lb, Ub, Nvar : Integer;
9815:                      ConsTerm : Boolean;
9816:                      B : TVector;
9817:                      V : TMatrix); external 'dmath';
9818: { Multiple linear regression by Gauss-Jordan method }
9819: procedure WMulFit(X : TMatrix;
9820:                      Y, S : TVector;
9821:                      Lb, Ub, Nvar : Integer;
9822:                      ConsTerm : Boolean;
9823:                      B : TVector;
9824:                      V : TMatrix); external 'dmath';
9825: { Weighted multiple linear regression by Gauss-Jordan method }
9826: procedure SVDFit(X : TMatrix;
9827:                      Y : TVector;
9828:                      Lb, Ub, Nvar : Integer;
9829:                      ConsTerm : Boolean;
9830:                      SVDTol : Float;
9831:                      B : TVector;
9832:                      V : TMatrix); external 'dmath';
9833: { Multiple linear regression by singular value decomposition }
9834: procedure WSVDFit(X : TMatrix;
9835:                      Y, S : TVector;
9836:                      Lb, Ub, Nvar : Integer;
9837:                      ConsTerm : Boolean;
9838:                      SVDTol : Float;
9839:                      B : TVector;
9840:                      V : TMatrix); external 'dmath';
9841: { Weighted multiple linear regression by singular value decomposition }
9842: procedure PolFit(X, Y : TVector;
9843:                      Lb, Ub, Deg : Integer;
9844:                      B : TVector;
9845:                      V : TMatrix); external 'dmath';
9846: { Polynomial regression by Gauss-Jordan method }
9847: procedure WPolFit(X, Y, S : TVector;
9848:                      Lb, Ub, Deg : Integer;
9849:                      B : TVector;
9850:                      V : TMatrix); external 'dmath';
9851: { Weighted polynomial regression by Gauss-Jordan method }
9852: procedure SVDPolFit(X, Y : TVector;
9853:                      Lb, Ub, Deg : Integer;
9854:                      SVDTol : Float;
9855:                      B : TVector;
9856:                      V : TMatrix); external 'dmath';
9857: { Unweighted polynomial regression by singular value decomposition }
9858: procedure WSVDPolFit(X, Y, S : TVector;
9859:                      Lb, Ub, Deg : Integer;
9860:                      SVDTol : Float;
9861:                      B : TVector;
9862:                      V : TMatrix); external 'dmath';
9863: { Weighted polynomial regression by singular value decomposition }
9864: procedure RegTest(Y, Ycalc : TVector;
9865:                      LbY, UbY : Integer;
9866:                      V : TMatrix;
9867:                      LbV, UbV : Integer;
9868:                      var Test : TRegTest); external 'dmath';
9869: { Test of unweighted regression }
9870: procedure WRegTest(Y, Ycalc, S : TVector;
9871:                      LbY, UbY : Integer;
9872:                      V : TMatrix;
9873:                      LbV, UbV : Integer;
9874:                      var Test : TRegTest); external 'dmath';
9875: { Test of weighted regression }
9876: { -----
9877: Nonlinear regression

```

```

9878: ----- }
9879: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9880: { Sets the optimization algorithm for nonlinear regression }
9881: function GetOptAlgo : TOptAlgo; external 'dmath';
9882: { Returns the optimization algorithm }
9883: procedure SetMaxParam(N : Byte); external 'dmath';
9884: { Sets the maximum number of regression parameters for nonlinear regression }
9885: function GetMaxParam : Byte; external 'dmath';
9886: { Returns the maximum number of regression parameters for nonlinear regression }
9887: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9888: { Sets the bounds on the I-th regression parameter }
9889: procedure GetParamBounds(I : Byte; var ParamMin, ParamMax:Float); external 'dmath';
9890: { Returns the bounds on the I-th regression parameter }
9891: procedure NLFit(RegFunc : TRegFunc;
9892:                     DerivProc : TDervProc;
9893:                     X, Y : TVector;
9894:                     Lb, Ub : Integer;
9895:                     MaxIter : Integer;
9896:                     Tol : Float;
9897:                     B : TVector;
9898:                     FirstPar,
9899:                     LastPar : Integer;
9900:                     V : TMatrix); external 'dmath';
9901: { Unweighted nonlinear regression }
9902: procedure WNLFit(RegFunc : TRegFunc;
9903:                     DerivProc : TDervProc;
9904:                     X, Y, S : TVector;
9905:                     Lb, Ub : Integer;
9906:                     MaxIter : Integer;
9907:                     Tol : Float;
9908:                     B : TVector;
9909:                     FirstPar,
9910:                     LastPar : Integer;
9911:                     V : TMatrix); external 'dmath';
9912: { Weighted nonlinear regression }
9913: procedure SetMCFile(FileName : String); external 'dmath';
9914: { Set file for saving MCMC simulations }
9915: procedure SimFit(RegFunc : TRegFunc;
9916:                     X, Y : TVector;
9917:                     Lb, Ub : Integer;
9918:                     B : TVector;
9919:                     FirstPar,
9920:                     LastPar : Integer;
9921:                     V : TMatrix); external 'dmath';
9922: { Simulation of unweighted nonlinear regression by MCMC }
9923: procedure WSimFit(RegFunc : TRegFunc;
9924:                     X, Y, S : TVector;
9925:                     Lb, Ub : Integer;
9926:                     B : TVector;
9927:                     FirstPar,
9928:                     LastPar : Integer;
9929:                     V : TMatrix); external 'dmath';
9930: { Simulation of weighted nonlinear regression by MCMC }
9931: { -----
9932: Nonlinear regression models
9933: ----- }

9934: procedure FracFit(X, Y : TVector;
9935:                     Lb, Ub : Integer;
9936:                     Deg1, Deg2 : Integer;
9937:                     ConsTerm : Boolean;
9938:                     MaxIter : Integer;
9939:                     Tol : Float;
9940:                     B : TVector;
9941:                     V : TMatrix); external 'dmath';
9942: { Unweighted fit of rational fraction }
9943: procedure WFracFit(X, Y, S : TVector;
9944:                     Lb, Ub : Integer;
9945:                     Deg1, Deg2 : Integer;
9946:                     ConsTerm : Boolean;
9947:                     MaxIter : Integer;
9948:                     Tol : Float;
9949:                     B : TVector;
9950:                     V : TMatrix); external 'dmath';
9951: { Weighted fit of rational fraction }

9952: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9953: { Returns the value of the rational fraction at point X }
9954: procedure ExpFit(X, Y : TVector;
9955:                     Lb, Ub, Nexp : Integer;
9956:                     ConsTerm : Boolean;
9957:                     MaxIter : Integer;
9958:                     Tol : Float;
9959:                     B : TVector;
9960:                     V : TMatrix); external 'dmath';
9961: { Unweighted fit of sum of exponentials }
9962: procedure WExpFit(X, Y, S : TVector;
9963:                     Lb, Ub, Nexp : Integer;
9964:                     ConsTerm : Boolean;
9965:                     MaxIter : Integer;
9966: 
```

```

9967:          Tol      : Float;
9968:          B       : TVector;
9969:          V       : TMatrix); external 'dmath';
9970: { Weighted fit of sum of exponentials }
9971: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9972: { Returns the value of the regression function at point X }
9973: procedure IncExpFit(X, Y      : TVector;
9974:                         Lb, Ub   : Integer;
9975:                         ConsTerm : Boolean;
9976:                         MaxIter  : Integer;
9977:                         Tol      : Float;
9978:                         B       : TVector;
9979:                         V       : TMatrix); external 'dmath';
9980: { Unweighted fit of model of increasing exponential }
9981: procedure WIIncExpFit(X, Y, S : TVector;
9982:                         Lb, Ub   : Integer;
9983:                         ConsTerm : Boolean;
9984:                         MaxIter  : Integer;
9985:                         Tol      : Float;
9986:                         B       : TVector;
9987:                         V       : TMatrix); external 'dmath';
9988: { Weighted fit of increasing exponential }
9989: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9990: { Returns the value of the regression function at point X }
9991: procedure ExplnFit(X, Y      : TVector;
9992:                         Lb, Ub   : Integer;
9993:                         MaxIter  : Integer;
9994:                         Tol      : Float;
9995:                         B       : TVector;
9996:                         V       : TMatrix); external 'dmath';
9997: { Unweighted fit of the "exponential + linear" model }
9998: procedure WExplnFit(X, Y, S : TVector;
9999:                         Lb, Ub   : Integer;
10000:                        MaxIter : Integer;
10001:                        Tol     : Float;
10002:                        B      : TVector;
10003:                        V      : TMatrix); external 'dmath';
10004: { Weighted fit of the "exponential + linear" model }
10005:
10006: function ExplnFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10007: { Returns the value of the regression function at point X }
10008: procedure MichFit(X, Y      : TVector;
10009:                         Lb, Ub   : Integer;
10010:                        MaxIter : Integer;
10011:                        Tol     : Float;
10012:                        B      : TVector;
10013:                        V      : TMatrix); external 'dmath';
10014: { Unweighted fit of Michaelis equation }
10015: procedure WMichFit(X, Y, S : TVector;
10016:                         Lb, Ub   : Integer;
10017:                         MaxIter : Integer;
10018:                         Tol     : Float;
10019:                         B      : TVector;
10020:                         V      : TMatrix); external 'dmath';
10021: { Weighted fit of Michaelis equation }
10022: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10023: { Returns the value of the Michaelis equation at point X }
10024: procedure MintFit(X, Y      : TVector;
10025:                         Lb, Ub   : Integer;
10026:                         MintVar : TMintVar;
10027:                         Fit_S0  : Boolean;
10028:                         MaxIter : Integer;
10029:                         Tol     : Float;
10030:                         B      : TVector;
10031:                         V      : TMatrix); external 'dmath';
10032: { Unweighted fit of the integrated Michaelis equation }
10033: procedure WMintFit(X, Y, S : TVector;
10034:                         Lb, Ub   : Integer;
10035:                         MintVar : TMintVar;
10036:                         Fit_S0  : Boolean;
10037:                         MaxIter : Integer;
10038:                         Tol     : Float;
10039:                         B      : TVector;
10040:                         V      : TMatrix); external 'dmath';
10041: { Weighted fit of the integrated Michaelis equation }
10042: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10043: { Returns the value of the integrated Michaelis equation at point X }
10044: procedure HillFit(X, Y      : TVector;
10045:                         Lb, Ub   : Integer;
10046:                         MaxIter : Integer;
10047:                         Tol     : Float;
10048:                         B      : TVector;
10049:                         V      : TMatrix); external 'dmath';
10050: { Unweighted fit of Hill equation }
10051: procedure WHillFit(X, Y, S : TVector;
10052:                         Lb, Ub   : Integer;
10053:                         MaxIter : Integer;
10054:                         Tol     : Float;
10055:                         B      : TVector;

```

```

10056:           V      : TMatrix); external 'dmath';
10057: { Weighted fit of Hill equation }
10058: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10059: { Returns the value of the Hill equation at point X }
10060: procedure LogiFit(X, Y      : TVector;
10061:                      Lb, Ub   : Integer;
10062:                      ConstTerm : Boolean;
10063:                      General  : Boolean;
10064:                      MaxIter  : Integer;
10065:                      Tol     : Float;
10066:                      B       : TVector;
10067:                      V       : TMatrix); external 'dmath';
10068: { Unweighted fit of logistic function }
10069: procedure WLogiFit(X, Y, S : TVector;
10070:                      Lb, Ub   : Integer;
10071:                      ConstTerm : Boolean;
10072:                      General  : Boolean;
10073:                      MaxIter  : Integer;
10074:                      Tol     : Float;
10075:                      B       : TVector;
10076:                      V       : TMatrix); external 'dmath';
10077: { Weighted fit of logistic function }
10078: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10079: { Returns the value of the logistic function at point X }
10080: procedure PKFit(X, Y      : TVector;
10081:                      Lb, Ub   : Integer;
10082:                      MaxIter  : Integer;
10083:                      Tol     : Float;
10084:                      B       : TVector;
10085:                      V       : TMatrix); external 'dmath';
10086: { Unweighted fit of the acid-base titration curve }
10087: procedure WPKFit(X, Y, S : TVector;
10088:                      Lb, Ub   : Integer;
10089:                      MaxIter  : Integer;
10090:                      Tol     : Float;
10091:                      B       : TVector;
10092:                      V       : TMatrix); external 'dmath';
10093: { Weighted fit of the acid-base titration curve }
10094: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10095: { Returns the value of the acid-base titration function at point X }
10096: procedure PowFit(X, Y      : TVector;
10097:                      Lb, Ub   : Integer;
10098:                      MaxIter  : Integer;
10099:                      Tol     : Float;
10100:                     B       : TVector;
10101:                     V       : TMatrix); external 'dmath';
10102: { Unweighted fit of power function }
10103: procedure WPowFit(X, Y, S : TVector;
10104:                      Lb, Ub   : Integer;
10105:                      MaxIter  : Integer;
10106:                      Tol     : Float;
10107:                      B       : TVector;
10108:                      V       : TMatrix); external 'dmath';
10109: { Weighted fit of power function }
10110:
10111: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10112: { Returns the value of the power function at point X }
10113: procedure GammaFit(X, Y      : TVector;
10114:                      Lb, Ub   : Integer;
10115:                      MaxIter  : Integer;
10116:                      Tol     : Float;
10117:                      B       : TVector;
10118:                      V       : TMatrix); external 'dmath';
10119: { Unweighted fit of gamma distribution function }
10120: procedure WGammaFit(X, Y, S : TVector;
10121:                      Lb, Ub   : Integer;
10122:                      MaxIter  : Integer;
10123:                      Tol     : Float;
10124:                      B       : TVector;
10125:                      V       : TMatrix); external 'dmath';
10126: { Weighted fit of gamma distribution function }
10127: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10128: { Returns the value of the gamma distribution function at point X }
10129: {
10130:   Principal component analysis
10131:   -----
10132:   procedure VecMean(X      : TMatrix;
10133:                      Lb, Ub, Nvar : Integer;
10134:                      M       : TVector); external 'dmath';
10135: { Computes the mean vector M from matrix X }
10136: procedure VecSD(X      : TMatrix;
10137:                      Lb, Ub, Nvar : Integer;
10138:                      M, S       : TVector); external 'dmath';
10139: { Computes the vector of standard deviations S from matrix X }
10140: procedure MatVarCov(X      : TMatrix;
10141:                      Lb, Ub, Nvar : Integer;
10142:                      M       : TVector;
10143:                      V       : TMatrix); external 'dmath';
10144: { Computes the variance-covariance matrix V from matrix X }

```

```

10145: procedure MatCorrel(V      : TMatrix;
10146:                      Nvar   : Integer;
10147:                      R      : TMatrix); external 'dmath';
10148: { Computes the correlation matrix R from the var-cov matrix V }
10149: procedure PCA(R       : TMatrix;
10150:                    Nvar   : Integer;
10151:                    Lambda : TVector;
10152:                    C, Rc  : TMatrix); external 'dmath';
10153: { Performs a principal component analysis of the correlation matrix R }
10154: procedure ScaleVar(X      : TMatrix;
10155:                        Lb, Ub, Nvar : Integer;
10156:                        M, S     : TVector;
10157:                        Z      : TMatrix); external 'dmath';
10158: { Scales a set of variables by subtracting means and dividing by SD's }
10159: procedure PrinFac(Z      : TMatrix;
10160:                        Lb, Ub, Nvar : Integer;
10161:                        C, F     : TMatrix); external 'dmath';
10162: { Computes principal factors }
10163: { -----
10164:   Strings
10165: ----- }

10166: function LTrim(S : String) : String; external 'dmath';
10167: { Removes leading blanks }
10168: function RTrim(S : String) : String; external 'dmath';
10169: { Removes trailing blanks }
10170: function Trim(S : String) : String; external 'dmath';
10171: { Removes leading and trailing blanks }
10172: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10173: { Returns a string made of character C repeated N times }
10174: function RFill(S : String; L : Byte) : String; external 'dmath';
10175: { Completes string S with trailing blanks for a total length L }
10176: function LFill(S : String; L : Byte) : String; external 'dmath';
10177: { Completes string S with leading blanks for a total length L }
10178: function CFill(S : String; L : Byte) : String; external 'dmath';
10179: { Centers string S on a total length L }
10180: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10181: { Replaces in string S all the occurrences of C1 by C2 }
10182: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10183: { Extracts a field from a string }
10184: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10185: { Parses a string into its constitutive fields }
10186: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10187: { Sets the numeric format }
10188: function FloatToStr(X : Float) : String; external 'dmath';
10189: { Converts a real to a string according to the numeric format }
10190: function IntToStr(N : LongInt) : String; external 'dmath';
10191: { Converts an integer to a string }
10192: function CompStr(Z : Complex) : String; external 'dmath';
10193: { Converts a complex number to a string }
10194: {$IFDEF DELPHI}
10195: function StrDec(S : String) : String; external 'dmath';
10196: { Set decimal separator to the symbol defined in SysUtils }
10197: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10198: { Test if a string represents a number and returns it in X }
10199: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10200: { Reads a floating point number from an Edit control }
10201: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10202: { Writes a floating point number in a text file }
10203: {$ENDIF}
10204: { -----
10205:   BGI / Delphi graphics
10206: ----- }

10207: function InitGraphics
10208: {$IFDEF DELPHI}
10209: (Width, Height : Integer) : Boolean;
10210: {$ELSE}
10211: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10212: { Enters graphic mode }
10213: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10214:                         X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10215: { Sets the graphic window }
10216: procedure SetOxScale(Scale           : TScale;
10217:                         OXMin, OXMax, OXStep : Float); external 'dmath';
10218: { Sets the scale on the Ox axis }
10219: procedure SetOyScale(Scale           : TScale;
10220:                         OYMin, OYMax, OYStep : Float); external 'dmath';
10221: { Sets the scale on the Oy axis }
10222: procedure GetOxScale(var Scale           : TScale;
10223:                         var OXMin, OXMax, OXStep : Float); external 'dmath';
10224: { Returns the scale on the Ox axis }
10225: procedure GetOyScale(var Scale           : TScale;
10226:                         var OYMin, OYMax, OYStep : Float); external 'dmath';
10227: { Returns the scale on the Oy axis }
10228: procedure SetGraphTitle>Title : String); external 'dmath'; { Sets the title for the graph }
10229: procedure SetOxTitle>Title : String); external 'dmath'; { Sets the title for the Ox axis }
10230: procedure SetOyTitle>Title : String); external 'dmath'; { Sets the title for the Oy axis }
10231: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10232: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10233: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }

```

```

10234: {$IFNDEF DELPHI}
10235: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10236: { Sets the font for the main graph title }
10237: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10238: { Sets the font for the Ox axis (title and labels) }
10239: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10240: { Sets the font for the Oy axis (title and labels) }
10241: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10242: { Sets the font for the legends }
10243: procedure SetClipping(Clip : Boolean); external 'dmath';
10244: { Determines whether drawings are clipped at the current viewport
10245: boundaries, according to the value of the Boolean parameter Clip }
10246: {$ENDIF}
10247: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10248: { Plots the horizontal axis }
10249: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10250: { Plots the vertical axis }
10251: procedure PlotGrid{$IFDEF DELPHI}(Canvas:TCanvas){$ENDIF} Grid:TGrid); external 'dmath';
10252: { Plots a grid on the graph }
10253: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10254: { Writes the title of the graph }
10255: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10256: { Sets the maximum number of curves and re-initializes their parameters }
10257: procedure SetPointParam
10258: {$IFDEF DELPHI}
10259: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10260: {$ELSE}
10261: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10262: { Sets the point parameters for curve # CurvIndex }
10263: procedure SetLineParam
10264: {$IFDEF DELPHI}
10265: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10266: {$ELSE}
10267: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10268: { Sets the line parameters for curve # CurvIndex }
10269: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10270: { Sets the legend for curve # CurvIndex }
10271: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10272: { Sets the step for curve # CurvIndex }
10273: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10274: procedure GetPointParam
10275: {$IFDEF DELPHI}
10276: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10277: {$ELSE}
10278: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10279: { Returns the point parameters for curve # CurvIndex }
10280: procedure GetLineParam
10281: {$IFDEF DELPHI}
10282: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10283: {$ELSE}
10284: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10285: { Returns the line parameters for curve # CurvIndex }
10286: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10287: { Returns the legend for curve # CurvIndex }
10288: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10289: { Returns the step for curve # CurvIndex }
10290: {$IFDEF DELPHI}
10291: procedure PlotPoint(Canvas : TCanvas;
10292: X, Y : Float; CurvIndex : Integer); external 'dmath';
10293: {$ELSE}
10294: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10295: {$ENDIF}
10296: { Plots a point on the screen }
10297: procedure PlotCurve{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10298: X, Y : TVector;
10299: Lb, Ub, CurvIndex : Integer); external 'dmath';
10300: { Plots a curve }
10301: procedure PlotCurveWithErrorBars{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10302: X, Y, S : TVector;
10303: Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10304: { Plots a curve with error bars }
10305: procedure PlotFunc{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10306: Func : TFunc;
10307: Xmin, Xmax : Float;
10308: {$IFDEF DELPHI}Npt : Integer;{$ENDIF}
10309: CurvIndex : Integer); external 'dmath';
10310: { Plots a function }
10311: procedure WriteLegend{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10312: NCurv : Integer;
10313: ShowPoints, ShowLines : Boolean); external 'dmath';
10314: { Writes the legends for the plotted curves }
10315: procedure ConRec{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}
10316: Nx, Ny, Nc : Integer;
10317: X, Y, Z : TVector;
10318: F : TMatrix); external 'dmath';
10319: { Contour plot }
10320: function Xpixel(X : Float):Integer; external 'dmath'; { Converts user abscissa X to screen coordinate }
10321: function Ypixel(Y : Float):Integer; external 'dmath'; { Converts user ordinate Y to screen coordinate }
10322: function Xuser(X : Integer):Float; external 'dmath'; { Converts screen coordinate X to user abscissa }

```

```

10323: function Yuser(Y : Integer):Float; external 'dmath'; {Converts screen coordinate Y to user ordinate }
10324: {$IFNDEF DELPHI}
10325: procedure LeaveGraphics; external 'dmath';
10326: { Quits graphic mode }
10327: {$ENDIF}
10328: { -----
10329:   LaTeX graphics
10330:   ----- }
10331: function TeX_InitGraphics(FileName      : String; PgWidth, PgHeight : Integer;
10332:                           Header       : Boolean) : Boolean; external 'dmath';
10333: { Initializes the LaTeX file }
10334: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10335: { Sets the graphic window }
10336: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10337: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10338: { Sets the scale on the Ox axis }
10339: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10340: { Sets the scale on the Oy axis }
10341: procedure TeX_SetGraphTitle>Title : String); external 'dmath'; { Sets the title for the graph }
10342: procedure TeX_SetOxTitle>Title : String); external 'dmath'; { Sets the title for the Ox axis }
10343: procedure TeX_SetOyTitle>Title : String); external 'dmath'; { Sets the title for the Oy axis }
10344: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10345: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10346: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10347: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10348: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10349: { Sets the maximum number of curves and re-initializes their parameters }
10350: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10351: { Sets the point parameters for curve # CurvIndex }
10352: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10353:                           Width : Float; Smooth : Boolean); external 'dmath';
10354: { Sets the line parameters for curve # CurvIndex }
10355: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10356: { Sets the legend for curve # CurvIndex }
10357: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10358: { Sets the step for curve # CurvIndex }
10359: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10360: { Plots a curve }
10361: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10362:                                       Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10363: { Plots a curve with error bars }
10364: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10365:                          Npt : Integer; CurvIndex : Integer); external 'dmath';
10366: { Plots a function }
10367: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10368: { Writes the legends for the plotted curves }
10369: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix); external 'dmath';
10370: { Contour plot }
10371: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10372: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10373:
10374: //*****unit uPSI_SynPdf;
10375: Function RawUTF8ToPDFString( const Value : RawUTF8 ) : PDFString
10376: Function _DateToTimeToPdfDate( ADate : TDateTime ) : TPdfDate
10377: Function _PfdDateToDate( const AText : TPdfDate ) : TDateTime
10378: Function PdfRect( Left, Top, Right, Bottom : Single ) : TPdfRect;
10379: Function PdfRect1( const Box : TPdfBox ) : TPdfRect;
10380: Function PdfBox( Left, Top, Width, Height : Single ) : TPdfBox
10381: //Function _GetCharCount( Text : PAnsichar ) : integer
10382: //Procedure L2R( W : PWideChar; L : integer )
10383: Function PdfCoord( MM : single ) : integer
10384: Function CurrentPrinterPageSize : TPDFPaperSize
10385: Function CurrentPrinterRes : TPoint
10386: Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8 )
10387: Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer )
10388: Procedure GDICommentLink( MetaHandle: HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10389: Const ('Usp10', 'String 'usp10.dll
10390:   AddTypes('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10391:   'fcharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10392:   TScriptState_set', 'set of TScriptState_enum
10393: //*****procedure SIRRegister_PMrand(CL: TPPascalCompiler); //ParkMiller
10394:
10395: procedure SIRRegister_PMrand(CL: TPPascalCompiler);
10396: begin
10397:   Procedure PMrandomize( I : word)
10398:   Function PMrandom : longint
10399:   Function Rrand : extended
10400:   Function Irand( N : word ) : word
10401:   Function Brand( P : extended ) : boolean
10402:   Function Nrand : extended
10403: end;
10404:
10405: procedure SIRRegister_Spring_Cryptography_Utils(CL: TPPascalCompiler);
10406: begin
10407:   Function Endian( x : LongWord ) : LongWord
10408:   Function Endian64( x : Int64 ) : Int64
10409:   Function spRol( x : LongWord; y : Byte) : LongWord
10410:   Function spRor( x : LongWord; y : Byte) : LongWord
10411:   Function Ror64( x : Int64; y : Byte ) : Int64

```

```

10412: end;
10413:
10414: procedure SIRegister_MapReader(CL: TPSPPascalCompiler);
10415: begin
10416:   Procedure ClearModules
10417:   Procedure ReadMapFile( Fname : string )
10418:   Function AddressInfo( Address : dword ) : string
10419:   end;
10420:
10421: procedure SIRegister_LibTar(CL: TPSPPascalCompiler);
10422: begin
10423:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOwner'
10424:   + 'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWrite'
10425:   + 'teByOther, tpExecuteByOther )
10426:   TTarPermissions', 'set of TTarPermission
10427:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10428:   + 'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10429:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10430:   TTarModes', 'set of TTarMode
10431:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDate'
10432:   + 'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10433:   + 'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING; '
10434:   + 'ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10435:   + 'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10436:   SIRegister_TTarArchive(CL);
10437:   SIRegister_TTarWriter(CL);
10438:   Function PermissionString( Permissions : TTarPermissions ) : STRING
10439:   Function ConvertFilename( Filename : STRING ) : STRING
10440:   Function FileTimeGMT( FileName : STRING ) : TDateTime;
10441:   Function FileTimeGMT1( SearchRec : TSearchRec ) : TDateTime;
10442:   Procedure ClearDirRec( var DirRec : TTarDirRec )
10443: end;
10444:
10445:
10446: //*****unit uPSI_TlHelp32;
10447: procedure SIRegister_TlHelp32(CL: TPSPPascalCompiler);
10448: begin
10449:   Const('MAX_MODULE_NAME32','LongInt'( 255 );
10450:   Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD ) : THandle
10451:   Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001 );
10452:   Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002 );
10453:   Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004 );
10454:   Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008 );
10455:   Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000 );
10456:   tagHEAPLIST32', 'record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10457:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10458:   AddTypeS('THeapList32', 'tagHEAPLIST32
10459:   Const('HF32_DEFAULT','LongInt'( 1 );
10460:   Const('HF32_SHARED','LongInt'( 2 );
10461:   Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10462:   Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10463:   AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THHandle; dwAdr'
10464:   + 'ress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10465:   + 'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10466:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10467:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10468:   Const('LF32_FIXED','LongWord').SetUInt( $00000001 );
10469:   Const('LF32_FREE','LongWord').SetUInt( $00000002 );
10470:   Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004 );
10471:   Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD ) : BOOL
10472:   Function Heap32Next( var lphe : THeapEntry32 ) : BOOL
10473:   DWORD; var lpNumberOfBytesRead : DWORD ) : BOOL
10474:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10475:   + '2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10476:   + 'aPri : Longint; dwFlags : DWORD; end
10477:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10478:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10479:   Function Thread32First( hSnapshot : THHandle; var lppte : TThreadEntry32 ) : BOOL
10480:   Function Thread32Next( hSnapshot : THHandle; var lppte : TThreadEntry32 ) : BOOL
10481: end;
10482: Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042 );
10483: Const('EW_REBOOTSYSTEM','LongWord( $0043 );
10484: Const('EW_EXITANDEXECAPP','LongWord( $0044 );
10485: Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ) );
10486: Const('EWX_LOGOFF','LongInt'( 0 );
10487: Const('EWX_SHUTDOWN','LongInt'( 1 );
10488: Const('EWX_REBOOT','LongInt'( 2 );
10489: Const('EWX_FORCE','LongInt'( 4 );
10490: Const('EWX_POWEROFF','LongInt'( 8 );
10491: Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10 );
10492: Function GET_APPCOMMAND_LPARAM( const lParam : LongInt ) : Shortint
10493: Function GET_DEVICE_LPARAM( const lParam : LongInt ) : Word
10494: Function GET_MOUSEKEY_LPARAM( const lParam : LongInt ) : Word
10495: Function GET_FLAGS_LPARAM( const lParam : LongInt ) : Word
10496: Function GET_KEYSTATE_LPARAM( const lParam : LongInt ) : Word
10497: Function GetWindowWord( hWnd : HWND; nIndex : Integer ) : Word
10498: Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word ) : Word
10499: Function GetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
10500: Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint

```

```

10501: Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10502: Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10503: Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10504: Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10505: Function GetDesktopWindow : HWND
10506: Function GetParent( hWnd : HWND) : HWND
10507: Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10508: Function GetTopWindow( hWnd : HWND) : HWND
10509: Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10510: Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10511: //Delphi DFM
10512: Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10513: Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10514: procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10515: function GetHighlightersFilter(AHighlighters: TStringList): string;
10516: function GetHighlighterFromExt(AHighlighters: TStringList; Extension: string):TSynCustomHighlighter;
10517: Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10518: Function OpenIcon( hWnd : HWND) : BOOL
10519: Function CloseWindow( hWnd : HWND) : BOOL
10520: Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10521: Function SetWindowPos(hWnd:HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UInt) : BOOL
10522: Function IsWindowVisible( hWnd : HWND) : BOOL
10523: Function IsIconic( hWnd : HWND) : BOOL
10524: Function AnyPopup : BOOL
10525: Function BringWindowToTop( hWnd : HWND) : BOOL
10526: Function IsZoomed( hWnd : HWND) : BOOL
10527: Function IsWindow( hWnd : HWND) : BOOL
10528: Function IsMenu( hMenu : HMENU) : BOOL
10529: Function IsChild( hWndParent, hWnd : HWND) : BOOL
10530: Function DestroyWindow( hWnd : HWND) : BOOL
10531: Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10532: Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10533: Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10534: Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10535: Function IsWindowUnicode( hWnd : HWND) : BOOL
10536: Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10537: Function IsWindowEnabled( hWnd : HWND) : BOOL
10538:
10539: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10540: begin
10541:   const ('ShowSetupDialogOptLong','String '--setup
10542: PrimaryConfPathOptLong','String '--primary-config-path=
10543: PrimaryConfPathOptShort','String '--pcp=
10544: SecondaryConfPathOptLong','String '--secondary-config-path=
10545: SecondaryConfPathOptShort','String '--scp=
10546: NoSplashScreenOptLong','String '--no-splash-screen
10547: NoSplashScreenOptShort','String '--nsc
10548: StartedByStartLazarusOpt','String '--started-by-startlazarus
10549: SkipLastProjectOpt','String '--skip-last-project
10550: DebugLogOpt','string '--debug-log=
10551: DebugLogOptEnable','String '--debug-enable=
10552: LanguageOpt','String '--language=
10553: LazarusDirOpt','String '--lazarusdir=
10554: Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10555: Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10556: Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10557: Function IsHelpRequested : Boolean
10558: Function IsVersionRequested : boolean
10559: Function GetLanguageSpecified : string
10560: Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10561: Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10562: Procedure ParseNoGuiCmdlineParams
10563: Function ExtractCmdLineFilenames : TStrings
10564: end;
10565:
10566:
10567: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10568: begin
10569:   Function CompareFilenames( const Filenam1, Filenam2 : string) : integer
10570:   Function CompareFilenamesIgnoreCase( const Filenam1, Filenam2 : string) : integer
10571:   Function CompareFileExt( const Filenam, Ext : string; CaseSensitive : boolean) : integer;
10572:   Function CompareFileExt1( const Filenam, Ext : string) : integer;
10573:   Function CompareFilenameStarts( const Filenam1, Filenam2 : string) : integer
10574:   Function CompareFilenames(Filenam1:PChar;Len1:integer; Filenam2:PChar;Len2:integer):integer
10575:   Function CompareFilenamesP( Filenam1, Filenam2 : PChar; IgnoreCase : boolean) : integer
10576:   Function DirPathExists( DirectoryName : string) : boolean
10577:   Function DirectoryIsWritable( const DirectoryName : string) : boolean
10578:   Function ExtractFileNameOnly( const AFilename : string) : string
10579:   Function FilenameIsAbsolute( const Thefilename : string) : boolean
10580:   Function FilenameIsWinAbsolute( const Thefilename : string) : boolean
10581:   Function FilenameIsUnixAbsolute( const Thefilename : string) : boolean
10582:   Function ForceDirectory(DirectoryName : string) : boolean
10583:   Procedure CheckIfFileIsExecutable( const AFilename : string)
10584:   Procedure CheckIfFileIsSymlink( const AFilename : string)
10585:   Function FileIsText( const AFilename : string) : boolean
10586:   Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10587:   Function FilenameIsTrimmed( const Thefilename : string) : boolean
10588:   Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10589:   Function TrimFilename( const AFilename : string) : string

```

```

10590: Function ResolveDots( const AFilename : string ) : string
10591: Procedure ForcePathDelims( var FileName : string )
10592: Function GetForcedPathDelims( const FileName : string ) : String
10593: Function CleanAndExpandfilename( const Filename : string ) : string
10594: Function CleanAndExpandDirectory( const Filename : string ) : string
10595: Function TrimAndExpandfilename( const Filename : string; const BaseDir : string ) : string
10596: Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string ) : string
10597: Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10598: Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
AlwaysRequireSharedBaseFolder: Boolean) : string
10599: Function FileIsInPath( const Filename, Path : string ) : boolean
10600: Function AppendPathDelim( const Path : string ) : string
10601: Function ChompPathDelim( const Path : string ) : string
10602: Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string ) : string
10603: Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string ) : string
10604: Function MinimizeSearchPath( const SearchPath : string ) : string
10605: Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
(*Function FileExistsUTF8( const Filename : string ) : boolean
10607: Function FileAgeUTF8( const FileName : string ) : Longint
10608: Function DirectoryExistsUTF8( const Directory : string ) : Boolean
10609: Function ExpandFileNameUTF8( const FileName : string; BaseDir : string ) : string
10610: Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10611: Function FindNextUTF8( var Rslt : TSearchRec ) : Longint
10612: Procedure FindCloseUTF8( var F : TSearchrec )
10613: Function FileSetDateUTF8( const FileName : String; Age : Longint ) : Longint
10614: Function FileGetAttrUTF8( const FileName : String ) : Longint
10615: Function FileSetAttrUTF8( const Filename : String; Attr : longint ) : Longint
10616: Function DeleteFileUTF8( const FileName : String ) : Boolean
10617: Function RenameFileUTF8( const OldName, NewName : String ) : Boolean
10618: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean ) : String
10619: Function FileIsReadOnlyUTF8( const FileName : String ) : Boolean
10620: Function GetCurrentDirUTF8 : String
10621: Function SetCurrentDirUTF8( const NewDir : String ) : Boolean
10622: Function CreateDirUTF8( const NewDir : String ) : Boolean
10623: Function RemoveDirUTF8( const Dir : String ) : Boolean
10624: Function ForceDirectoriesUTF8( const Dir : string ) : Boolean
10625: Function FileOpenUTF8( const FileName : string; Mode : Integer ) : THandle
10626: Function FileCreateUTF8( const FileName : string ) : THandle;
10627: Function FileCreateUTF81( const FileName : string; Rights : Cardinal ) : THandle;
10628: Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal ) : THandle;
10629: Function FileSizeUtf8( const Filename : string ) : int64
10630: Function GetfileDescription( const AFilename : string ) : string
10631: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean ) : string
10632: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean ) : string
10633: Function GetTempFileNameUTF8( const Dir, Prefix : String ) : String*)
10634: Function IsUNCPath( const Path : String ) : Boolean
10635: Function ExtractUNCVolume( const Path : String ) : String
10636: Function ExtractFileRoot( FileName : String ) : String
10637: Function GetDarwinSystemFilename( Filename : string ) : string
10638: Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean )
10639: Function StrToCmdLineParam( const Param : string ) : string
10640: Function MergeCmdLineParams( ParamList : TStrings ) : string
10641: Procedure InvalidateFileStateCache( const Filename : string )
10642: Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList;
10643: Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean ) : TStringList
10644: Function FindAllDocs(const Root, extmask: string): TStringlist;
10645: Function ReadfileToString( const Filename : string ) : string
10646: procedure Incl(var X: longint; N: Longint);
10647:
10648: type
10649:   TCopyFileFlag = ( cffOverwriteFile,
10650:                      cffCreateDestDirectory, cffPreserveTime );
10651:   TCopyFileFlags = set of TCopyFileFlag;*
10652:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10653:   TCopyFileFlags', 'set of TCopyFileFlag
10654:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags ) : Boolean
10655: end;
10656:
10657: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10658: begin
10659:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10660:   SIRegister_TMask(CL);
10661:   SIRegister_TParseStringList(CL);
10662:   SIRegister_TMaskList(CL);
10663:   Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean ) : Boolean
10664:   Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean ) : Bool;
10665:   Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10666:   Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10667: end;
10668:
10669: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10670: begin
10671:   //PShellHookInfo', '^TShellHookInfo // will not work
10672:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10673:   SHELLHOOKINFO', 'TShellHookInfo
10674:   LPSHELLHOOKINFO', 'PShellHookInfo
10675:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10676:   SIRegister_TJvShellHook(CL);

```

```

10677: Function InitJvShellHooks : Boolean
10678: Procedure UnInitJvShellHooks
10679: end;
10680:
10681: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10682: begin
10683:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10684:   +', dcHasSelSel, dcWantTab, dcNative )
10685:   TDlgCodes', 'set of TDlgCode
10686:   'dcWantMessage', ' dcWantAllKeys);
10687:   SIRegister_IJVExControl(CL);
10688:   SIRegister_IJVDenySubClassing(CL);
10689:   SIRegister_TStructPtrMessage(CL);
10690:   Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10691:   Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10692:   Procedure DrawDotNetControl( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10693:   Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg: TMessage; MouseOver: Boolean; Color: TColor);
10694:   Function CreateWMMessages( Msg : Integer; WParam : Integer; LParam : Longint ) : TMessage;
10695:   Function CreateWMMessages1( Msg : Integer; WParam : Integer; LParam : TControl ) : TMessage;
10696:   Function SmallPointToLong( const Pt : TSmallPoint ) : Longint
10697:   Function ShiftStateToKeyData( Shift : TShiftState ) : Longint
10698:   Function GetFocusedControl( AControl : TControl ) : TWinControl
10699:   Function DlgcToDlgCodes( Value : Longint ) : TDlgCodes
10700:   Function DlgCodesToDlgc( Value : TDlgCodes ) : Longint
10701:   Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor );
10702:   Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage ) : Boolean
10703:   SIRegister_TJVExControl(CL);
10704:   SIRegister_TJVExWinControl(CL);
10705:   SIRegister_TJVExCustomControl(CL);
10706:   SIRegister_TJVExGraphicControl(CL);
10707:   SIRegister_TJVExHintWindow(CL);
10708:   SIRegister_TJVExPubGraphicControl(CL);
10709: end;
10710:
10711: (*-----*)
10712: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10713: begin
10714:   Procedure EncodeStream( Input, Output : TStream)
10715:   Procedure DecodeStream( Input, Output : TStream)
10716:   Function EncodeString1( const Input : string ) : string
10717:   Function DecodeString1( const Input : string ) : string
10718: end;
10719:
10720: (*-----*)
10721: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10722: begin
10723:   SIRegister_TWebAppRegInfo(CL);
10724:   SIRegister_TWebAppRegList(CL);
10725:   Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10726:   Procedure RegisterWebApp( const AFileName, AProgID : string)
10727:   Procedure UnregisterWebApp( const AProgID : string)
10728:   Function FindRegisteredWebApp( const AProgID : string ) : string
10729:   Function CreateRegistry( InitializeNewFile : Boolean ) : TCustomIniFile
10730:   'sUDPPort', 'String 'UDPPort
10731: end;
10732:
10733: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10734: begin
10735:   // TStringDynArray', 'array of string
10736:   Function GetEnvVarValue( const VarName : string ) : string
10737:   Function SetEnvVarValue( const VarName, VarValue : string ) : Integer
10738:   Function DeleteEnvVar( const VarName : string ) : Integer
10739:   Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
BufSize:Int):Int;
10740:   Function ExpandEnvVars( const Str : string ) : string
10741:   Function GetAllEnvVars( const Vars : TStrings ) : Integer
10742:   Procedure GetAllEnvVarNames( const Names : TStrings );
10743:   Function GetAllEnvVarNames1 : TStringDynArray;
10744:   Function EnvBlockSize : Integer
10745:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject )
10746:   SIRegister_TPJEnvVarsEnumerator(CL);
10747:   SIRegister_TPJEnvVars(CL);
10748:   FindClass('TOBJECT'), 'EPJEnvVars
10749:   FindClass('TOBJECT'), 'EPJEnvVars
10750:   //Procedure Register
10751: end;
10752:
10753: (*-----*)
10754: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10755: begin
10756:   'cOneSecInMS', 'LongInt'( 1000 );
10757:   // 'cDefTimeSlice', 'LongInt'( 50 );
10758:   // 'cDefMaxExecTime', 'cOneMinInMS';
10759:   'cAppErrorMask', 'LongInt'( 1 shl 29 );
10760:   Function IsApplicationError( const ErrCode : LongWord ) : Boolean
10761:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10762:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10763:   Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor ):TPJConsoleColors;
10764:   Function MakeConsoleColors1( const AForeground, ABackground : TColor ) : TPJConsoleColors;

```

```

10765: Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor ) : TPJConsoleColors;
10766: Function MakeSize( const ACX, ACY : LongInt ) : TSize
10767: SIRegister_TPJCustomConsoleApp(CL);
10768: SIRegister_TPJConsoleApp(CL);
10769: end;
10770:
10771: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10772: begin
10773:   INVALID_IP_ADDRESS', 'LongWord').SetUInt( $ffffffff );
10774:   t_encoding', '( uuencode, base64, mime )
10775:   Function internet_date( date : TDateTime ) : string
10776:   Function lookup_hostname( const hostname : string ) : longint
10777:   Function my_hostname : string
10778:   Function my_ip_address : longint
10779:   Function ip2string( ip_address : longint ) : string
10780:   Function resolve_hostname( ip : longint ) : string
10781:   Function address_from( const s : string; count : integer ) : string
10782:   Function encode_base64( data : TStream ) : TStringList
10783:   Function decode_base64( source : TStringList ) : TMemoryStream
10784:   Function posn( const s, t : string; count : integer ) : integer
10785:   Function poscn( c : char; const s : string; n : integer ) : integer
10786:   Function filename_of( const s : string ) : string
10787: //Function trim( const s : string ) : string
10788: //Procedure setlength( var s : string; l : byte )
10789: Function TimeZoneBias : longint
10790: Function eight2seven_quoteprint( const s : string ) : string
10791: Function eight2seven_german( const s : string ) : string
10792: Function seven2eight_quoteprint( const s : string ) : string end;
10793: type in_addr', 'record s_bytes : array[1..4] of byte; end;
10794: Function socketerror : cint
10795: Function fpsocket( domain : cint; xtype : cint; protocol : cint ) : cint
10796: Function fprecv( s : cint; buf : __pointer; len : size_t; flags : cint ) : ssize_t
10797: Function fpsend( s : cint; msg : __pointer; len : size_t; flags : cint ) : ssize_t
10798: //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen ) : cint
10799: Function fplisten( s : cint; backlog : cint ) : cint
10800: //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint ) : cint
10801: //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen ) : cint
10802: //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen ) : cint
10803: Function NetAddrToStr( Entry : in_addr ) : String
10804: Function HostAddrToStr( Entry : in_addr ) : String
10805: Function StrToHostAddr( IP : String ) : in_addr
10806: Function StrToNetAddr( IP : String ) : in_addr
10807: SOL_SOCKET', 'LongWord').SetUInt( $ffff );
10808:   cint8', 'shortint
10809:   cuint8', 'byte
10810:   cchar', 'cint8
10811:   cschar', 'cint8
10812:   cuchar', 'cuint8
10813:   cint16', 'smallint
10814:   cuint16', 'word
10815:   cshort', 'cint16
10816:   csshort', 'cint16
10817:   cushort', 'cuint16
10818:   cint32', 'longint
10819:   cuint32', 'longword
10820:   cint', 'cint32
10821:   csint', 'cint32
10822:   cuint', 'cuint32
10823:   csigned', 'cint
10824:   cunsigned', 'cuint
10825:   cint64', 'int64
10826:   clonglong', 'cint64
10827:   cslonglong', 'cint64
10828:   cbool', 'longbool
10829:   cfloat', 'single
10830:   cdouble', 'double
10831:   clongdouble', 'extended
10832:
10833: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10834: begin
10835:   TlkJSONTypes', '( jsBase, jsNumber, jsString, jsBoolean, jsNull, jsList, jsObject )
10836:   SIRegister_TlkJSONdotnetclass(CL);
10837:   SIRegister_TlkJSONbase(CL);
10838:   SIRegister_TlkJSONnumber(CL);
10839:   SIRegister_TlkJSONstring(CL);
10840:   SIRegister_TlkJSONboolean(CL);
10841:   SIRegister_TlkJSONnull(CL);
10842:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Ele : TlkJSONba'
10843:     + 'se; data : TObject; var Continue : Boolean)
10844:   SIRegister_TlkJSONcustomlist(CL);
10845:   SIRegister_TlkJSONlist(CL);
10846:   SIRegister_TlkJSONobjectmethod(CL);
10847:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10848:   TlkHashFunction', 'Function ( const ws : WideString ) : cardinal
10849:   SIRegister_TlkHashTable(CL);
10850:   SIRegister_TlkBalTree(CL);
10851:   SIRegister_TlkJSONObject(CL);
10852:   SIRegister_TlkJSON(CL);
10853:   SIRegister_TlkJSONstreamed(CL);

```

```

10854: Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10855: end;
10856:
10857: procedure SIRegister_ZSysUtils(CL: TPPSPascalCompiler);
10858: begin
10859:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10860:   SIRegister_TZSortedList(CL);
10861:   Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10862:   Function zLastDelimiter( const Delimiters, Str : string) : Integer
10863:   //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10864:   //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10865:   Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10866:   Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10867:   Function EndsWith1( const Str, SubStr : WideString) : Boolean;
10868:   Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10869:   Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10870:   Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10871:   Function SQLStrToFloat( const Str : AnsiString) : Extended
10872:   //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10873:   //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10874:   Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10875:   Function StrToBoolEx( Str : string) : Boolean
10876:   Function BoolToStrEx( Bool : Boolean) : String
10877:   Function IsIpAddr( const Str : string) : Boolean //IsIP()
10878:   Function zSplitString( const Str, Delimiters : string) : TStrings
10879:   Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10880:   Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10881:   Function ComposeString( List : TStrings; const Delimiter : string) : string
10882:   Function FloatToSQLStr( Value : Extended) : string
10883:   Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10884:   Function SplitStringEx( const Str, Delimiter : string) : TStrings
10885:   Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10886:   Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10887:   Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10888:   Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10889:   Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10890:   Function StrToBytes3( const Value : WideString) : TByteDynArray;
10891:   Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10892:   Function BytesToVar( const Value : TByteDynArray) : Variant
10893:   Function VarToBytes( const Value : Variant) : TByteDynArray
10894:   Function AnsiSQLDateToDate( const Value : string) : TDateTime
10895:   Function TimestampStrToDate( const Value : string) : TDateTime
10896:   Function DateToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10897:   Function EncodeCString( const Value : string) : string
10898:   Function DecodeCString( const Value : string) : string
10899:   Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10900:   Function MemPas( Buffer : PChar; Length : LongInt) : string
10901:   Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
SubVersion:Int);
10902:   Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
SubVersion:Integer);
10903:   Function FormatsSQLVersion( const SQLVersion : Integer) : String
10904:   Function ZStrToFloat( Value : AnsiChar) : Extended;
10905:   Function ZStrToFloat1( Value : AnsiString) : Extended;
10906:   Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10907:   Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10908:   Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10909:   Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10910:   Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10911:   Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10912:   Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10913: end;
10914:
10915: unit uPSI_ZEncoding;
10916: Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10917: Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10918: Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10919: Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10920: Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10921: Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10922: Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10923: Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10924: Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10925: Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10926: Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10927: Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10928: Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10929: Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10930: Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10931: Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP : Word) : UTF8String
10932: Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10933: Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP : Word) : AnsiString
10934: Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10935: Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10936: Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10937: Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10938: Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10939: Function ZConvertStringToUnicodeWithAutoEncode( const Src : String; const StringCP : Word) : WideString
10940: Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString

```

```

10941: Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10942: Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10943: Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10944: Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10945: Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10946: Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10947: Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10948: Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10949: Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10950: Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10951: Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10952: Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10953: Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10954: Function ZDefaultSystemCodePage : Word
10955: Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10956: function MPing(const AHost: string;const ATimes:integer; out AvgMS:Double):Boolean;
10958:
10959: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10960: begin
10961:   ('RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10962:   ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10963:   ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10964:   ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10965:   ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10966:   ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10967:   ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10968:   {'alDefault', '1 RPC_C_AUTHN_LEVEL_DEFAULT};
10969:   {'alNone', '2 RPC_C_AUTHN_LEVEL_NONE};
10970:   {'alConnect', '3 RPC_C_AUTHN_LEVEL_CONNECT};
10971:   {'alCall', '4 RPC_C_AUTHN_LEVEL_CALL};
10972:   {'alPacket', '5 RPC_C_AUTHN_LEVEL_PKT};
10973:   {'alPacketIntegrity', '6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY};
10974:   {'alPacketPrivacy', '7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY};
10975:   ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10976:   ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10977:   ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10978:   ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10979:   ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10980:   {'ilDefault', '0 RPC_C_IMP_LEVEL_DEFAULT};
10981:   {'ilAnonymous', '1 RPC_C_IMP_LEVEL_ANONYMOUS};
10982:   {'ilIdentiry', '2 RPC_C_IMP_LEVEL_IDENTIFY};
10983:   {'ilImpersonate', '3 RPC_C_IMP_LEVEL_IMPERSONATE};
10984:   {'ilDelegate', '4 RPC_C_IMP_LEVEL_DELEGATE};
10985:   ('EOAC_NONE','LongWord').SetUInt( $0);
10986:   ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10987:   ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10988:   ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10989:   ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10990:   ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10991:   ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10992:   ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10993:   ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10994:   ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10995:   FindClass('TOBJECT'),'EBoldCom
10996: Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10997: Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10998: Function BoldStreamToVariant( Stream : TStream) : OleVariant
10999: Function BoldStringsToVariant( Strings : TStrings) : OleVariant
11000: Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
11001: Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
11002: Function BoldVariantArrayOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
11003: Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
11004: Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
11005: Function BoldGetValue( Data : OleVariant; const Name : string) : OleVariant
11006: Procedure BoldSetValue( Data : OleVariant; const Name : string; Value : OleVariant)
11007: Function BoldCreateGUID : TGUID
11008: Function BoldCreateComObject( const ClsId, IID : TGUID; out Obj : variant; out Res : HResult) : Boolean
11009: Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
Res:HRes):Bool;
11010: Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
11011: Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
11012: end;
11013:
11014: (*-----*)
11015: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11016: begin
11017:   Function ParseISODate( s : string) : TDateTime
11018:   Function ParseISODateTime( s : string) : TDateTime
11019:   Function ParseISOTime( str : string) : TDateTime
11020: end;
11021:
11022: (*-----*)
11023: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11024: begin
11025:   Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
11026:   Function BoldCreateGUIDWithBracketsAsString : string
11027: end;
11028:

```

```

11029: procedure SIRегистер_BoldFileHandler(CL: TPSPPascalCompiler);
11030: begin
11031:   FindClass('TOBJECT'), 'TBoldFileHandler'
11032:   FindClass('TOBJECT'), 'TBoldDiskFileHandler'
11033:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11034:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList )
11035:   SIRегистер_TBoldFileHandler(CL);
11036:   SIRегистер_TBoldDiskFileHandler(CL);
11037:   Procedure BoldCloseAllFilehandlers
11038:   Procedure BoldRemoveUnchangedFilesFromEditor
11039:   Function BoldFileHandlerList : TBoldObjectArray
11040:   Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
11041:   OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
11042: end;
11043: procedure SIRегистер_BoldWinINet(CL: TPSPPascalCompiler);
11044: begin
11045:   PCharArr', 'array of PChar
11046:   Function BoldInternetOpen(Agent:String;
11047:   AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr;
11048:   Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11049:   NumberofBytesRead:Card):LongBool;
11050:   Function BoldInternetCloseHandle( HINet : Pointer ) : LongBool
11051:   Cardinal; Reserved : Cardinal) : LongBool
11052:   Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
11053:   Cardinal; Context : Cardinal ) : LongBool
11054:   Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
11055:   : PCharArr; Flags, Context : Cardinal) : Pointer
11056:   Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
11057:   Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11058:   Function BoldInternetAttemptConnect( dwReserved : DWORD ) : DWORD
11059:   Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
11060:   Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11061:   Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11062: end;
11063: procedure SIRегистер_BoldQueryUserDlg(CL: TPSPPascalCompiler);
11064: begin
11065:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11066:   SIRегистер_TfrmBoldQueryUser(CL);
11067:   Function QueryUser( const Title, Query : string ) : TBoldQueryResult
11068: (*-----*)
11069: procedure SIRегистер_BoldQueue(CL: TPSPPascalCompiler);
11070: begin
11071:   //('befIsInDisplayList',' BoldElementFlag0 );
11072:   //('befStronglyDependedOfPrioritized',' BoldElementFlag1 );
11073:   //('befFollowerSelected',' BoldElementFlag2 );
11074:   FindClass('TOBJECT'), 'TBoldQueue
11075:   FindClass('TOBJECT'), 'TBoldQueueable
11076:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11077:   SIRегистер_TBoldQueueable(CL);
11078:   Function BoldQueueFinalized : Boolean
11079:   Function BoldInstalledQueue : TBoldQueue
11080: end;
11081: procedure SIRегистер_Barcod(CL: TPSPPascalCompiler);
11082: begin
11083:   const mmPerInch', 'Extended').setExtended( 25.4 );
11084:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,
11085:   + 'bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11086:   + 'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11087:   + 'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11088:   + 'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11089:   TBarLineType', '( white, black, black_half )
11090:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11091:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st
11092:   + 'pBottomLeft, stpBottomRight, stpBottomCenter )
11093:   TCheckSumMethod', '( csmNone, csmModulolo10 )
11094:   SIRегистер_TAsBarcode(CL);
11095:   Function CheckSumModulolo10( const data : string ) : string
11096:   Function ConvertMmToPixelsX( const Value : Double ) : Integer
11097:   Function ConvertMmToPixelsY( const Value : Double ) : Integer
11098:   Function ConvertInchToPixelsX( const Value : Double ) : Integer
11099:   Function ConvertInchToPixelsY( const Value : Double ) : Integer
11100: end;
11101: procedure SIRегистер_Geometry(CL: TPSPPascalCompiler); //OpenGL
11102: begin
11103:   THomogeneousByteVector', 'array[0..3] of Byte
11104:   THomogeneousWordVector', 'array[0..3] of Word
11105:   THomogeneousIntVector', 'array[0..3] of Integer
11106:   THomogeneousFltVector', 'array[0..3] of single
11107:   THomogeneousDblVector', 'array[0..3] of double
11108:   THomogeneousExtVector', 'array[0..3] of extended

```

```

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

```

```

11200: Function MakeAffineVector( V : array of Single ) : TAffineVector
11201: Function MakeQuaternion( Imag : array of Single; Real : Single ) : TQuaternion
11202: Function MakeVector( V : array of Single ) : TVectorGL
11203: Function PointInPolygonGL( xp, yp : array of Single; x, y : Single ) : Boolean
11204: Function VectorAffineDblToFlt( V : TAffineDblVector ) : TAffineVector
11205: Function VectorDblToFlt( V : THomogeneousDblVector ) : THomogeneousVector
11206: Function VectorAffineFltToDbl( V : TAffineVector ) : TAffineDblVector
11207: Function VectorFltToDbl( V : TVectorGL ) : THomogeneousDblVector
11208: Function ArcCosGL( X : Extended ) : Extended
11209: Function ArcSingL( X : Extended ) : Extended
11210: Function ArcTan2GL( Y, X : Extended ) : Extended
11211: Function CoTanGL( X : Extended ) : Extended
11212: Function DegToRadGL( Degrees : Extended ) : Extended
11213: Function RadToDegGL( Radians : Extended ) : Extended
11214: Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended )
11215: Function TanGL( X : Extended ) : Extended
11216: Function Turn( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11217: Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11218: Function Pitch( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11219: Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11220: Function Roll( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11221: Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11222: end;
11223:
11224:
11225: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11226: begin
11227:   Function RegCreateKey( const RootKey : HKEY; const Key, Value : string ) : Longint
11228:   Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string ) : Boolean
11229:   Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string ) : Boolean
11230:   Function RegReadBool( const RootKey : HKEY; const Key, Name : string ) : Boolean
11231:   Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean ) : Boolean
11232:   Function RegReadInteger( const RootKey : HKEY; const Key, Name : string ) : Integer
11233:   Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer ) : Integer
11234:   Function RegReadString( const RootKey : HKEY; const Key, Name : string ) : string
11235:   Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string ) : string
11236:   Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string ) : Int64
11237:   Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64 ) : Int64
11238:   Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean )
11239:   Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer )
11240:   Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string )
11241:   Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64 )
11242:   Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings ) : Boolean
11243:   Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings ) : Boolean
11244:   Function RegHasSubKeys( const RootKey : HKEY; const Key : string ) : Boolean
11245:   Function RegKeyExists( const RootKey : HKEY; const Key : string ) : Boolean
11246:   AddTypeS('TExecKind', ['ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11247:     +'RunOnce, ekServiceRun, ekServiceRunOnce ')
11248:   AddClassN(FindClass('TOBJECT'), EJclRegistryError
11249:   Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string ) : Boolean
11250:   Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string ) : Boolean
11251:   Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
11252: Items:TStrings):Bool;
11253:   Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
11254: SaveTo:TStrings):Bool;
11255:   Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11256: end;
11257:
11258: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11259: begin
11260:   CLSID_StdComponentCategoriesMgr', 'GUID '{0002E005-0000-0000-C000-00000000046}
11261:   CATID_SafeForInitializing', 'GUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11262:   CATID_SafeForScripting', 'GUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11263:   icMAX_CATEGORY_DESC_LEN', 'LongInt'( 128 );
11264:   FindClass('TOBJECT'), 'EInvalidParam
11265:   Function IsDCOMInstalled : Boolean
11266:   Function IsDCOMEnabled : Boolean
11267:   Function GetDCOMVersion : string
11268:   Function GetMDACVersion : string
11269:   Function GetMDACVersion2 : string
11270:   Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
11271: VarArray:OleVariant):HRESULT;
11272:   Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HRESULT;
11273:   Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
11274: VarArray:OleVariant):HRESULT;
11275:   Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HRESULT;
11276:   Function MarshalInterMachineInterfaceInVarArray( const iid:TIID;unk:IUnknown;var
11277: VarArray:OleVariant):HRESULT;
11278:   Function CreateComponentCategory( const CatID : TGUID; const sDescription : string ) : HRESULT
11279:   Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HRESULT
11280:   Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HRESULT
11281:   Function ResetIStreamToStart( Stream : IStream ) : Boolean
11282:   Function SizeOfIStreamContents( Stream : IStream ) : Largeint
11283:   Function StreamToVariantArray( Stream : TStream ) : OleVariant;
11284:   Function StreamToVariantArray1( Stream : IStream ) : OleVariant;
11285:   Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream );
11286:   Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream );
11287: end;
11288:
```

```

11284:
11285: procedure SIRegister_JclUnitConv_mX2(CL: TPSCompiler);
11286: begin
11287:   Const ('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11288:   FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11289:   KelvinFreezingPoint','Extended').setExtended( 273.15);
11290:   CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11291:   FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11292:   KelvinAbsoluteZero','Extended').setExtended( 0.0);
11293:   DegPerCycle','Extended').setExtended( 360.0);
11294:   DegPerGrad','Extended').setExtended( 0.9);
11295:   DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11296:   GradPerCycle','Extended').setExtended( 400.0);
11297:   GradPerDeg','Extended').setExtended( 1.11111111111111111111111111111111);
11298:   GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11299:   RadPerCycle ','Extended').setExtended( 6.283185307179586476925286766559);
11300:   RadPerDeg ','Extended').setExtended( 0.017453292519943295769236907684886);
11301:   RadPerGrad ','Extended').setExtended( 0.015707963267948966192313216916398);
11302:   CyclePerDeg ','Extended').setExtended( 0.0027777777777777777777777777777778);
11303:   CyclePerGrad ','Extended').setExtended( 0.0025);
11304:   CyclePerRad ','Extended').setExtended( 0.15915494309189533576888376337251);
11305:   ArcMinutesPerDeg ','Extended').setExtended( 60.0);
11306:   ArcSecondsPerArcMinute ','Extended').setExtended( 60.0);
11307:   Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11308:   Function MakePercentage( const Step, Max : Longint) : Longint
11309:   Function CelsiusToKelvin( const T : double) : double
11310:   Function CelsiusToFahrenheit( const T : double) : double
11311:   Function KelvinToCelsius( const T : double) : double
11312:   Function KelvinToFahrenheit( const T : double) : double
11313:   Function FahrenheitToCelsius( const T : double) : double
11314:   Function FahrenheitToKelvin( const T : double) : double
11315:   Function CycleToDeg( const Cycles : double) : double
11316:   Function CycleToGrad( const Cycles : double) : double
11317:   Function CycleToRad( const Cycles : double) : double
11318:   Function DegToCycle( const Degrees : double) : double
11319:   Function DegToGrad( const Degrees : double) : double
11320:   Function DegToRad( const Degrees : double) : double
11321:   Function GradToCycle( const Grads : double) : double
11322:   Function GradToDeg( const Grads : double) : double
11323:   Function GradToRad( const Grads : double) : double
11324:   Function RadToCycle( const Radians : double) : double
11325:   Function RadToDeg( const Radians : double) : double
11326:   Function RadToGrad( const Radians : double) : double
11327:   Function DmsToDeg( const D, M : Integer; const S : double) : double
11328:   Function DmsToRad( const D, M : Integer; const S : double) : double
11329:   Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11330:   Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11331:   Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11332:   Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11333:   Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11334:   Procedure CartesianToSpherical( const X, Y, Z : double; out Rho, Phi, Theta : double)
11335:   Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11336:   Procedure SphericalToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11337:   Function CmToInch( const Cm : double) : double
11338:   Function InchToCm( const Inch : double) : double
11339:   Function FeetToMetre( const Feet : double) : double
11340:   Function MetreToFeet( const Metre : double) : double
11341:   Function YardToMetre( const Yard : double) : double
11342:   Function MetreToYard( const Metre : double) : double
11343:   Function NmToKm( const Nm : double) : double
11344:   Function KmToNm( const Km : double) : double
11345:   Function KmToSm( const Km : double) : double
11346:   Function SmToKm( const Sm : double) : double
11347:   Function LitreToGalUs( const Litre : double) : double
11348:   Function GalUsToLitre( const GalUs : double) : double
11349:   Function GalUsToGalCan( const GalUs : double) : double
11350:   Function GalCanToGalUs( const GalCan : double) : double
11351:   Function GalUsToGalUk( const GalUs : double) : double
11352:   Function GalUkToGalUs( const GalUk : double) : double
11353:   Function LitreToGalCan( const Litre : double) : double
11354:   Function GalCanToLitre( const GalCan : double) : double
11355:   Function LitreToGalUk( const Litre : double) : double
11356:   Function GalUkToLitre( const GalUk : double) : double
11357:   Function KgToLb( const Kg : double) : double
11358:   Function LbToKg( const Lb : double) : double
11359:   Function KgToOz( const Kg : double) : double
11360:   Function OzToKg( const Oz : double) : double
11361:   Function CwtUsToKg( const Cwt : double) : double
11362:   Function CwtUkToKg( const Cwt : double) : double
11363:   Function KaratToKg( const Karat : double) : double
11364:   Function KgToCwtUs( const Kg : double) : double
11365:   Function KgToCwtUk( const Kg : double) : double
11366:   Function KgToKarat( const Kg : double) : double
11367:   Function KgToSton( const Kg : double) : double
11368:   Function KgToLton( const Kg : double) : double
11369:   Function StonToKg( const STon : double) : double
11370:   Function LtonToKg( const Lton : double) : double
11371:   Function QrUsToKg( const Qr : double) : double
11372:   Function QrUkToKg( const Qr : double) : double

```

```

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

```

```

11459: Function FindAuthClass( AuthName : String ) : TIIdAuthenticationClass
11460: GBase64CodeTable', 'string'ABCDEFGHJKLMNOPQRSTUWXZYabdefghijklmnopqrstuvwxyz0123456789+/
11461: GXECodeTable', 'string'+-0123456789ABCDEFGHJKLMNOPQRSTUWXZYabdefghijklmnopqrstuvwxyz[ \]^_
11462: GUUECodeTable', 'string'!"#$%&'"()*,-./0123456789:<=>?@ABCDEFGHIJKLMNPQRSTUWXZY[\]^_
11463: end;
11464:
11465: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11466: begin
11467: ('IpAny', 'LongWord').SetUInt( $00000000 );
11468: IpLoopBack', 'LongWord').SetUInt( $7F000001 );
11469: IpBroadcast', 'LongWord').SetUInt( $FFFFFF );
11470: IpNone', 'LongWord').SetUInt( $FFFFFF );
11471: PortAny', 'LongWord( $0000 );
11472: SocketMaxConnections', 'LongInt'( 5 );
11473: TIpAddr', 'LongWord
11474: TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11475: Function HostToNetLong( HostLong : LongWord ) : LongWord
11476: Function HostToNetShort( HostShort : Word ) : Word
11477: Function NetToHostLong( NetLong : LongWord ) : LongWord
11478: Function NetToHostShort( NetShort : Word ) : Word
11479: Function StrToIp( Ip : string ) : TIpAddr
11480: Function IpToStr( Ip : TIpAddr ) : string
11481: end;
11482:
11483: (*-----*)
11484: procedure SIRegister_ALSMTPCClient(CL: TPSPascalCompiler);
11485: begin
11486: TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11487: +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11488: +'amSha1, AlsmtpClientAuthAutoSelect )
11489: TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11490: SIRegister_TAlSmtpClient(CL);
11491: end;
11492:
11493: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11494: begin
11495: 'TBitNo', 'Integer
11496: TStByteNo', 'Integer
11497: TStationNo', 'Integer
11498: TInOutNo', 'Integer
11499: TIO', '( EE, AA, NE, NA )
11500: TBitSet', 'set of TBitNo
11501: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11502: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11503: TBitAddr', 'LongInt
11504: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11505: TByteAddr', 'SmallInt
11506: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11507: Function BitAddr(aIo: TIO; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11508: Function BusbitAddr(aIo:TIO;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11509: Procedure BitAddrToValues(aBitAddr:TBitAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte;var aBitNo:TBitNo);
11510: Function BitAddrToStr( Value : TBitAddr ) : string
11511: Function StrToBitAddr( const Value : string ) : TBitAddr
11512: Function ByteAddr( aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte ) : TByteAddr
11513: Function BusByteAddr(aIo:TIO;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11514: Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte)
11515: Function ByteAddrToStr( Value : TByteAddr ) : string
11516: Function StrToByteAddr( const Value : string ) : TByteAddr
11517: Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer )
11518: Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer )
11519: Function InOutStateToStr( State : TInOutState ) : string
11520: Function MasterErrorToStr( ErrorCode : TErrorCode ) : string
11521: Function SlaveErrorToStr( ErrorCode : TErrorCode ) : string
11522: end;
11523:
11524: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11525: begin
11526: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048,
11527: +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11528: DpmiPmVector', 'Int64
11529: 'DInterval', 'LongInt'( 1000 );
11530: //'DEnabled', 'Boolean')BoolToStr( True );
11531: 'DIntFreq', 'String' if64
11532: // 'DMessages', 'Boolean if64);
11533: SIRegister_TwdxCustomTimer(CL);
11534: SIRegister_TwdxTimer(CL);
11535: SIRegister_TwdxRtcTimer(CL);
11536: SIRegister_TCustomIntTimer(CL);
11537: SIRegister_TIntTimer(CL);
11538: SIRegister_TRtcIntTimer(CL);
11539: Function RealNow : TDateTime
11540: Function MsToDateTIme( Millisecond : LongInt ) : TDateTime
11541: Function DateTImeToMs( Time : TDateTime ) : LongInt
11542: end;
11543:
11544: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11545: begin
11546: TIIdSyslogPRI', 'Integer

```

```

11547: TIIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11548:   +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11549:   +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11550:   +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11551:   +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )'
11552: TIIdSyslogSeverity', '( slEmergency, slAlert, slCritical, slError, slWarning, slNotice, slInformational, slDebug)'
11553: SIRegister_TIdSysLogMsgPart(CL);
11554: SIRegister_TIdSysLogMessage(CL);
11555: Function FacilityToString( AFac : TIIdSyslogFacility ) : string
11556: Function SeverityToString( ASec : TIIdSyslogSeverity ) : string
11557: Function NoToSeverity( ASev : Word ) : TIIdSyslogSeverity
11558: Function logSeverityToNo( ASev : TIIdSyslogSeverity ) : Word
11559: Function NoToFacility( AFac : Word ) : TIIdSyslogFacility
11560: Function logFacilityToNo( AFac : TIIdSyslogFacility ) : Word
11561: end;
11562:
11563: procedure SIRegister_TextUtils(CL: TPSPPascalCompiler);
11564: begin
11565:   'UWhitespace', 'String' '(?:\s*)'
11566:   Function StripSpaces( const AText : string ) : string
11567:   Function CharCount( const AText : string; Ch : Char ) : Integer
11568:   Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11569:   Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11570: end;
11571:
11572:
11573: procedure SIRegister_ExtPascalUtils(CL: TPSPPascalCompiler);
11574: begin
11575:   ExtPascalVersion', 'String' '0.9.8
11576:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11577:     +'Opera, brKonqueror, brMobileSafari )'
11578:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )'
11579:   AddTypeS('TExtProcedure', 'Procedure
11580:   Function DetermineBrowser( const UserAgentStr : string ) : TBrowser
11581:   Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11582:   Function ExtExplode( Delim : char; const S : string; Separator : char ) : TStringList
11583:   Function FirstDelimiter( const Delimiters, S : string; Offset : integer ) : integer
11584:   Function RPosEx( const Substr, Str : string; Offset : integer ) : integer
11585:   Function CountStr( const Substr, Str : string; UntilStr : string ) : integer
11586:   Function StrToJS( const S : string; UseBR : boolean ) : string
11587:   Function CaseOf( const S : string; const Cases : array of string ) : integer
11588:   Function RCaseOf( const S : string; const Cases : array of string ) : integer
11589:   Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer ) : string
11590:   Function SetPaddings(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11591:   Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11592:   Function ExtBefore( const BeforeS, AfterS, S : string ) : boolean
11593:   Function IsUpperCase( S : string ) : boolean
11594:   Function BeautifyJS( const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11595:   Function BeautifyCSS( const AStyle : string ) : string
11596:   Function LengthRegExp( Rex : string; CountAll : Boolean ) : integer
11597:   Function JSDateToDate( JSDate : string ) : TDate
11598: end;
11599:
11600: procedure SIRegister_JclShell(CL: TPSPPascalCompiler);
11601: begin
11602:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )'
11603:   TSHDeleteOptions', 'set of TSHDeleteOption
11604:   TSHRenameOption', '( roSilent, roRenameOnCollision )'
11605:   TSHRenameOptions', 'set of TSHRenameOption
11606:   Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions ) : Boolean
11607:   Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions ) : Boolean
11608:   Function SHRenamefile( const Src, Dest : string; Options : TSHRenameOptions ) : Boolean
11609:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )'
11610:   TEnumFolderFlags', 'set of TEnumFolderFlag
11611:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11612:     +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11613:     +'IEnumIdList; Folder : IShellFolder; end
11614:   Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11615:   Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11616:   Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11617:   Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11618:   Function GetSpecialFolderLocation( const Folder : Integer) : string
11619:   Function DisplayPropDialog( const Handle : HWND; const FileName : string ) : Boolean;
11620:   Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList ) : Boolean;
11621:   Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint ) : Boolean
11622:   Function OpenFolder( const Path : string; Parent : HWND ) : Boolean
11623:   Function OpenSpecialFolder( FolderID : Integer; Parent : HWND ) : Boolean
11624:   Function SHReallocMem( var P : Pointer; Count : Integer ) : Boolean
11625:   Function SHAllocMem( out P : Pointer; Count : Integer ) : Boolean
11626:   Function SHGetMem( var P : Pointer; Count : Integer ) : Boolean
11627:   Function SHFreeMem( var P : Pointer ) : Boolean
11628:   Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder ) : PItemIdList
11629:   Function PathToPidl( const Path : string; Folder : IShellFolder ) : PItemIdList
11630:   Function PathToPidlBind( const FileName : string; out Folder : IShellFolder ) : PItemIdList
11631:   Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11632:   Function PidlCompare( const Pid1, Pid2 : PItemIdList ) : Boolean
11633:   Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList ) : Boolean
11634:   Function PidlFree( var IdList : PItemIdList ) : Boolean
11635:   Function PidlGetDepth( const Pidl : PItemIdList ) : Integer

```

```

11636: Function PidlGetLength( const Pidl : PIItemIdList ) : Integer
11637: Function PidlGetNext( const Pidl : PIItemIdList ) : PIItemIdList
11638: Function PidlToPath( IdList : PIItemIdList ) : string
11639: Function StrRetFreeMem( StrRet : TStrRet ) : Boolean
11640: Function StrRetToString( IdList : PIItemIdList; StrRet : TStrRet; Free : Boolean ) : string
11641: PShellLink', '^TShellLink // will not work'
11642: TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11643: +'ingDirectory : string; IdList : PIItemIdList; Target : string; Description '
11644: +' : string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11645: Procedure ShellLinkFree( var Link : TShellLink )
11646: Function ShellLinkResolve( const FileName : string; var Link : TShellLink ) : HRESULT
11647: Function ShellLinkCreate( const Link : TShellLink; const FileName : string ) : HRESULT
11648: Function ShellLinkCreateSystem( const Link:TShellLink;const Folder:Integer; const FileName:string ):HRESULT;
11649: Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon ) : Boolean
11650: Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo ) : Boolean
11651: Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal ) : HICON
11652: Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean ) : Boolean
11653: Function OverlayIconShortCut( var Large, Small : HICON ) : Boolean
11654: Function OverlayIconShared( var Large, Small : HICON ) : Boolean
11655: Function SHGetFileInfo( const Folder : IShellFolder; Item : PIItemIdList ) : string
11656: Function ShellExecEx( const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int ):Bool;
11657: Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int ):Bool;
11658: Function ShellExecAndWait( const FileName:string;const Params: string;const Verb:string;CmdShow:Int ):Bool;
11659: Function ShellOpenAs( const FileName : string ) : Boolean
11660: Function ShellRasodial( const EntryName : string ) : Boolean
11661: Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer ):Boolean
11662: Function GetFileNameIcon( const FileName : string; Flags : Cardinal ) : HICON
11663: TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )'
11664: Function GetfileExeType( const FileName : TfileName ) : TJclFileExeType
11665: Function ShellFindExecutable( const FileName, DefaultDir : string ) : string
11666: Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD )
11667: Function OemKeyScan( wOemChar : Word ) : DWORD
11668: Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD )
11669: end;
11670:
11671: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11672: begin
11673: xmlVersion', 'String '1.0 FindClass('TOBJECT'),'Exml
11674: //Function xmlValidChar( const Ch : AnsiChar ) : Boolean
11675: Function xmlValidChar1( const Ch : UCS4Char ) : Boolean;
11676: Function xmlValidChar2( const Ch : WideChar ) : Boolean;
11677: Function xmlIsSpaceChar( const Ch : WideChar ) : Boolean
11678: Function xmlIsLetter( const Ch : WideChar ) : Boolean
11679: Function xmlIsDigit( const Ch : WideChar ) : Boolean
11680: Function xmlIsNameStartChar( const Ch : WideChar ) : Boolean
11681: Function xmlIsNameChar( const Ch : WideChar ) : Boolean
11682: Function xmlIsPubidChar( const Ch : WideChar ) : Boolean
11683: Function xmlValidName( const Text : UnicodeString ) : Boolean
11684: //xmlSpace', 'Char #$20 or #$9 or #$D or #$A';
11685: //Function xmlSkipSpace( var P : PWideChar ) : Boolean
11686: //Function xmlSkipBq( var P : PWideChar ) : Boolean
11687: //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString ) : Boolean
11688: //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer )
11689: : TUnicodeCodecClass
11690: Function xmlResolveEntityReference( const RefName : UnicodeString ) : WideChar
11691: Function xmlTag( const Tag : UnicodeString ) : UnicodeString
11692: Function xmlEndTag( const Tag : UnicodeString ) : UnicodeString
11693: Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString ) : UnicodeString
11694: Function xmlEmptyTag( const Tag, Attr : UnicodeString ) : UnicodeString
11695: Procedure xmlSafeTextInPlace( var Txt : UnicodeString )
11696: Function xmlSafeText( const Txt : UnicodeString ) : UnicodeString
11697: Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer ) : UnicodeString
11698: Function xmlTabIndent( const IndentLevel : Integer ) : UnicodeString
11699: Function xmlComment( const Comment : UnicodeString ) : UnicodeString
11700: Procedure SelfTestcXMLFunctions
11701: end;
11702: (*-----*)
11703: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11704: begin
11705: Function AWaitCursor : IUnknown
11706: Function ChangeCursor( NewCursor : TCursor ) : IUnknown
11707: Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean )
11708: Function YesNo( const ACaption, AMsg : string ) : boolean
11709: Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings )
11710: Function GetBorlandLibPath( Version : integer; ForDelphi : boolean ) : string
11711: Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean ) : string
11712: Procedure GetPathlist( Version : integer; ForDelphi : boolean; Strings : TStrings )
11713: Procedure GetSystemPaths( Strings : TStrings )
11714: Procedure MakeEditNumeric( EditHandle : integer )
11715: end;
11716:
11717: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11718: begin
11719: AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11720: 'BI_YUY2','LongWord( $3259559 );
11721: 'BI_UYVY','LongWord').SetUInt( $59565955 );
11722: 'BI_BTYUV','LongWord').SetUInt( $50313459 );
11723: 'BI_YVU9','LongWord').SetUInt( $39555659 );

```

```

11724: 'BI_YUV12','LongWord( $30323449);
11725: 'BI_Y8','LongWord').SetUInt( $20203859);
11726: 'BI_Y211','LongWord').SetUInt( $31313259);
11727: Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec;
11728: Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11729: end;
11730:
11731: (*-----*)
11732: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11733: begin
11734: 'WM_USER','LongWord').SetUInt( $0400);
11735: 'WM_CAP_START','LongWord').SetUInt($0400);
11736: 'WM_CAP_END','longword').SetUInt($0400+85);
11737: //WM_CAP_START+ 85
11738: // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11739: Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11740: Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11741: Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11742: Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11743: Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11744: Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11745: Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11746: Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11747: Function capGetUserData( hwnd : THandle) : LongInt
11748: Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11749: Function capDriverDisconnect( hwnd : THandle) : LongInt
11750: Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11751: Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11752: Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11753: Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11754: Function capfileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11755: Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11756: Function capfileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11757: Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11758: Function capfileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11759: Function capEditCopy( hwnd : THandle) : LongInt
11760: Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11761: Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11762: Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11763: Function capDlgVideoFormat( hwnd : THandle) : LongInt
11764: Function capDlgVideoSource( hwnd : THandle) : LongInt
11765: Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11766: Function capDlgVideoCompression( hwnd : THandle) : LongInt
11767: Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11768: Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11769: Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11770: Function capPreview( hwnd : THandle; f : Word) : LongInt
11771: Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11772: Function capOverlay( hwnd : THandle; f : Word) : LongInt
11773: Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11774: Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11775: Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11776: Function capGrabFrame( hwnd : THandle) : LongInt
11777: Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11778: Function capCaptureSequence( hwnd : THandle) : LongInt
11779: Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11780: Function capCaptureStop( hwnd : THandle) : LongInt
11781: Function capCaptureAbort( hwnd : THandle) : LongInt
11782: Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11783: Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11784: Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11785: Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11786: Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11787: Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11788: Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11789: Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11790: Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11791: Function capPalettePaste( hwnd : THandle) : LongInt
11792: Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11793: Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11794: //PCapDriverCaps', '^TCapDriverCaps // will not work
11795: TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11796: +' ; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11797: +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hvid'
11798: +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11799: //PCapStatus', '^TCapStatus // will not work
11800: TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11801: +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOINT'
11802: +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11803: +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11804: +'rrentWaveSamples : DWORD; dwCurrentTimeLapsedMS : DWORD; hPalCurrent : HP'
11805: +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD; '
11806: +' wNumAudioAllocated : WORD; end
11807: //PCaptureParms', '^TCaptureParms // will not work
11808: TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11809: +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11810: +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11811: +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11812: +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'

```

```

11813: +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11814: +'wMCISearchBar : WORD; dwMCISearchTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11815: +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11816: +'he : BOOL; AVStreamMaster : WORD; end
11817: // PCapInfoChunk, '^TCapInfoChunk // will not work
11818: //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11819: 'CONTROLCALLBACK_PREROLL', 'LongInt'( 1);
11820: 'CONTROLCALLBACK_CAPTURING', 'LongInt'( 2);
11821: Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
: Integer; hwndParent : THandle; nID : Integer ) : THandle
11822: Function
capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11823: 'IDS_CAP_BEGIN', 'LongInt'( 300);
11824: 'IDS_CAP_END', 'LongInt'( 301);
11825: 'IDS_CAP_INFO', 'LongInt'( 401);
11826: 'IDS_CAP_OUTOFMEM', 'LongInt'( 402);
11827: 'IDS_CAP_FILEEXISTS', 'LongInt'( 403);
11828: 'IDS_CAP_ERRORPALOPEN', 'LongInt'( 404);
11829: 'IDS_CAP_ERRORPALSOLVE', 'LongInt'( 405);
11830: 'IDS_CAP_ERRORDIBSAVE', 'LongInt'( 406);
11831: 'IDS_CAP_DEFAVIEXT', 'LongInt'( 407);
11832: 'IDS_CAP_DEFPALEXT', 'LongInt'( 408);
11833: 'IDS_CAP_CANTOPEN', 'LongInt'( 409);
11834: 'IDS_CAP_SEQ_MSGSTART', 'LongInt'( 410);
11835: 'IDS_CAP_SEQ_MSGSTOP', 'LongInt'( 411);
11836: 'IDS_CAP_VIDEEDITERR', 'LongInt'( 412);
11837: 'IDS_CAP_READONLYFILE', 'LongInt'( 413);
11838: 'IDS_CAP_WRITEERROR', 'LongInt'( 414);
11839: 'IDS_CAP_NODISKSPACE', 'LongInt'( 415);
11840: 'IDS_CAP_SETFILESIZE', 'LongInt'( 416);
11841: 'IDS_CAP_SAVEASPERCENT', 'LongInt'( 417);
11842: 'IDS_CAP_DRIVER_ERROR', 'LongInt'( 418);
11843: 'IDS_CAP_WAVE_OPEN_ERROR', 'LongInt'( 419);
11844: 'IDS_CAP_WAVE_ALLOC_ERROR', 'LongInt'( 420);
11845: 'IDS_CAP_WAVE_PREPARE_ERROR', 'LongInt'( 421);
11846: 'IDS_CAP_WAVE_ADD_ERROR', 'LongInt'( 422);
11847: 'IDS_CAP_WAVE_SIZE_ERROR', 'LongInt'( 423);
11848: 'IDS_CAP_VIDEO_OPEN_ERROR', 'LongInt'( 424);
11849: 'IDS_CAP_VIDEO_ALLOC_ERROR', 'LongInt'( 425);
11850: 'IDS_CAP_VIDEO_PREPARE_ERROR', 'LongInt'( 426);
11851: 'IDS_CAP_VIDEO_ADD_ERROR', 'LongInt'( 427);
11852: 'IDS_CAP_VIDEO_SIZE_ERROR', 'LongInt'( 428);
11853: 'IDS_CAP_FILE_OPEN_ERROR', 'LongInt'( 429);
11854: 'IDS_CAP_FILE_WRITE_ERROR', 'LongInt'( 430);
11855: 'IDS_CAP_RECORDING_ERROR', 'LongInt'( 431);
11856: 'IDS_CAP_RECORDING_ERROR2', 'LongInt'( 432);
11857: 'IDS_CAP_AVI_INIT_ERROR', 'LongInt'( 433);
11858: 'IDS_CAP_NO_FRAME_CAP_ERROR', 'LongInt'( 434);
11859: 'IDS_CAP_NO_PALETTE_WARN', 'LongInt'( 435);
11860: 'IDS_CAP_MCI_CONTROL_ERROR', 'LongInt'( 436);
11861: 'IDS_CAP_MCI_CANT_STEP_ERROR', 'LongInt'( 437);
11862: 'IDS_CAP_NO_AUDIO_CAP_ERROR', 'LongInt'( 438);
11863: 'IDS_CAP_AVI_DRAWDIB_ERROR', 'LongInt'( 439);
11864: 'IDS_CAP_COMPRESSOR_ERROR', 'LongInt'( 440);
11865: 'IDS_CAP_AUDIO_DROP_ERROR', 'LongInt'( 441);
11866: 'IDS_CAP_STAT_LIVE_MODE', 'LongInt'( 500);
11867: 'IDS_CAP_STAT_OVERLAY_MODE', 'LongInt'( 501);
11868: 'IDS_CAP_STAT_CAP_INIT', 'LongInt'( 502);
11869: 'IDS_CAP_STAT_CAP_FINI', 'LongInt'( 503);
11870: 'IDS_CAP_STAT_PALETTE_BUILD', 'LongInt'( 504);
11871: 'IDS_CAP_STAT_OPTPAL_BUILD', 'LongInt'( 505);
11872: 'IDS_CAP_STAT_I_FRAMES', 'LongInt'( 506);
11873: 'IDS_CAP_STAT_L_FRAMES', 'LongInt'( 507);
11874: 'IDS_CAP_STAT_CAP_L_FRAMES', 'LongInt'( 508);
11875: 'IDS_CAP_STAT_CAP_AUDIO', 'LongInt'( 509);
11876: 'IDS_CAP_STAT_VIDEOCURRENT', 'LongInt'( 510);
11877: 'IDS_CAP_STAT_VIDEOAUDIO', 'LongInt'( 511);
11878: 'IDS_CAP_STAT_VIDEOONLY', 'LongInt'( 512);
11879: 'IDS_CAP_STAT_FRAMESDROPPED', 'LongInt'( 513);
11880: 'AVICAP32', 'String 'AVICAP32.dll
11881: end;
11882:
11883: procedure SIRegister_ALFcnnMisc(CL: TPSPascalCompiler);
11884: begin
11885: Function AlBoolToInt( Value : Boolean ) : Integer
11886: Function ALMediumPos( LTotal, LBorder, LObject : integer ) : Integer
11887: Function AlIsValidEmail( const Value : AnsiString ) : boolean
11888: Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime ) : TdateTime
11889: Function ALInc( var x : integer; Count : integer ) : Integer
11890: function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11891: function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11892: procedure ALSaveStringToFile(Str: AnsiString; filename: AnsiString);
11893: Function AlIsInteger(const S: AnsiString): Boolean;
11894: function AlIsDecimal(const S: AnsiString): boolean;
11895: Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11896: function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11897: function ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''): AnsiString;
11898: function ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11899: function ALUTF8removeBOM(const S: AnsiString): AnsiString;

```

```

11900: Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11901: Function ALRandomStr(const aLength: Longint): AnsiString;
11902: Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11903: Function ALRandomStrU(const aLength: Longint): String;
11904: end;
11905:
11906: procedure SIRегистер_ALJSONDoc(CL: TPSPPascalCompiler);
11907: begin
11908:   Procedure ALJSONToTStrings(const AJsonStr: AnsiString; aLst: TALStrings; const aNullStr: AnsiString; const
11909:     aTrueStr: AnsiString; const aFalseStr: AnsiString)
11910:   end;
11911:   procedure SIRегистер_ALWindows(CL: TPSPPascalCompiler);
11912:   begin
11913:     _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11914:       +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11915:       +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11916:       +'; ullAvailExtendedVirtual : Int64; end
11917:     TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11918:     Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX ) : BOOL
11919:     Function ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG ) : LONGLONG
11920:     'INVALID_SET_FILE_POINTER', 'LongInt'( DWORD( - 1 ) );
11921:     'QUOTA_LIMITS_HARDWS_MIN_DISABLE', 'LongWord').SetUInt( $2 );
11922:     'QUOTA_LIMITS_HARDWS_MIN_ENABLE', 'LongWord').SetUInt( $1 );
11923:     'QUOTA_LIMITS_HARDWS_MAX_DISABLE', 'LongWord').SetUInt( $8 );
11924:     'QUOTA_LIMITS_HARDWS_MAX_ENABLE', 'LongWord').SetUInt( $4 );
11925:   end;
11926:
11927:   procedure SIRегистер_IPCThrd(CL: TPSPPascalCompiler);
11928:   begin
11929:     SIRегистер_THandledObject(CL);
11930:     SIRегистер_TEvent(CL);
11931:     SIRегистер_TMutex(CL);
11932:     SIRегистер_TSharedMem(CL);
11933:     'TRACE_BUF_SIZE', 'LongInt'( 200 * 1024 );
11934:     'TRACE_BUFFER', 'String 'TRACE_BUFFER
11935:     'TRACE_MUTEX', 'String 'TRACE_MUTEX
11936:     //PTTraceEntry', 'TTTraceEntry // will not work
11937:     SIRегистер_TIPCTracer(CL);
11938:     'MAX_CLIENTS', 'LongInt'( 6 );
11939:     'IPC TIMEOUT', 'LongInt'( 2000 );
11940:     'IPCBUFFER_NAME', 'String 'BUFFER_NAME
11941:     'BUFFER_MUTEX_NAME', 'String 'BUFFER_MUTEX
11942:     'MONITOR_EVENT_NAME', 'String 'MONITOR_EVENT
11943:     'CLIENT_EVENT_NAME', 'String 'CLIENT_EVENT
11944:     'CONNECT_EVENT_NAME', 'String 'CONNECT_EVENT
11945:     'CLIENT_DIR_NAME', 'String 'CLIENT_DIRECTORY
11946:     'CLIENT_DIR_MUTEX', 'String 'DIRECTORY_MUTEX
11947:     FindClass('TOBJECT'), EMonitorActive
11948:     FindClass('TOBJECT'), TIPTThread
11949:     TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11950:       +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11951:       +'ach, evClientSwitch, evClientSignal, evClientExit )
11952:     TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11953:     TClientFlags', 'set of TClientFlag
11954:     //PEventData', 'TEventData // will not work
11955:     TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11956:       +'lag; Flags : TClientFlags; end
11957:     TConnectEvent', 'Procedure ( Sender : TIPTThread; Connecting : Boolean )
11958:     TDirUpdateEvent', 'Procedure ( Sender : TIPTThread )
11959:     TIPNotifyEvent', 'Procedure ( Sender : TIPTThread; Data : TEventData )
11960:     //PIPEEventInfo', '^TPIPEEventInfo // will not work
11961:     TPIPEEventInfo', 'record FID:Integer; FKind:TEventKind; FData:TEventData;end
11962:     SIRегистер_TIPCEvent(CL);
11963:     //PClientDirRecords', '^TClientDirRecords // will not work
11964:     SIRегистер_TClientDirectory(CL);
11965:     TIPOState', '( stInactive, stDisconnected, stConnected )
11966:     SIRегистер_TIPCThread(CL);
11967:     SIRегистер_TIPCMonitor(CL);
11968:     SIRегистер_TIPCCClient(CL);
11969:     Function IsMonitorRunning( var Hndl : THandle ) : Boolean
11970:   end;
11971:
11972: (*-----*)
11973: procedure SIRегистер_ALGSMComm(CL: TPSPPascalCompiler);
11974: begin
11975:   SIRегистер_TALGSMComm(CL);
11976:   Function ALGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString ) : AnsiString
11977:   Procedure ALGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
11978:     AMessage:AnsiString);
11979:   Function ALGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString ) : AnsiString
11980:   Function ALGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
11981:     UseGreekAlphabet:Bool):Widestring;
11982:   function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11983:   procedure SIRегистер_ALHttpCommon(CL: TPSPPascalCompiler);
11984:   begin
11985:     TALHTTPPropertyChangeEvent', 'Procedure(sender:Tobject;const PropertyIndex:Integer;

```

```

11986: TALHTTPProtocolVersion', '( HTTPPpv_1_0, HTTPPpv_1_1 )
11987: TALHTTPMethod','(HTTPPmt_Get,HTTPPmt_Post,HTTPPmt_Head,HTTPPmt_Trace,HTTPPmt_Put,HTTPPmt_Delete);
11988: TIInternetScheme', 'integer
11989: TALIPv6Binary', 'array[1..16] of Char;
11990: // TALIPv6Binary = array[1..16] of ansiChar;
11991: // TIInternetScheme = Integer;
11992: SIRегистrier_TALHTTPRequestHeader(CL);
11993: SIRегистrier_TALHTTPCookie(CL);
11994: SIRегистrier_TALHTTPCookieCollection(CL);
11995: SIRегистrier_TALHTTPResponseHeader(CL);
11996: Function ALHTTPDecode( const AStr : AnsiString ) : AnsiString
11997: Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings )
11998: // Procedure ALEXtractHTTPFields( Separators,WhiteSpace, Quotes:TSysCharSet,
Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11999: // Procedure ALEXtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
12000: // Procedure ALEXtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
12001: Function AlRemoveShemeFromUrl( aUrl : AnsiString ) : ansiString
12002: Function AlExtractShemeFromUrl( aUrl : AnsiString ) : TIInternetScheme
12003: Function AlExtractHostNameFromUrl( aUrl : AnsiString ) : AnsiString
12004: Function AlExtractDomainNameFromUrl( aUrl : AnsiString ) : AnsiString
12005: Function AlExtractUrlPathFromUrl( aUrl : AnsiString ) : AnsiString
12006: Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName, HostName, UserName, Password, UrlPath,
ExtraInfo : AnsiString; var PortNumber : integer ) : Boolean;
12007: Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password, UrlPath,
Anchor : AnsiString; Query : TALStrings; var PortNumber : integer ) : Boolean;
12008: Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
12009: Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString ) : AnsiString;
12010: Function AlRemoveAnchorFromUrl1( aUrl : AnsiString ) : AnsiString;
12011: Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString ) : AnsiString;
12012: Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
12013: Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
12014: Function ALDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
12015: Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime ) : Boolean
12016: Function ALRfc822StrToGMTDateTime( const s : AnsiString ) : TDateTime
12017: Function ALTryIPv4StrToNumeric( aIPv4Str : AnsiString; var aIPv4Num : Cardinal ) : Boolean
12018: Function ALIPv4StrToNumeric( aIPv4 : AnsiString ) : Cardinal
12019: Function ALNumericToIPv4Str( aIPv4 : Cardinal ) : AnsiString
12020: Function ALZeroIpV6 : TALIPv6Binary
12021: Function ALTryIPV6StrToBinary( aIPv6Str : AnsiString; var aIPv6Bin : TALIPv6Binary ) : Boolean
12022: Function ALIPv6StrTobinary( aIPv6 : AnsiString ) : TALIPv6Binary
12023: Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary ) : AnsiString
12024: Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : AnsiString ) : TALIPv6Binary
12025: end;
12026:
12027: procedure SIRегистrier_ALFcnHTML(CL: TPSPascalCompiler);
12028: begin
12029:   Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
DecodeHTMLText:Bool;
12030:   Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
12031:   Function ALXMLCDataElementEncode( Src : AnsiString ) : AnsiString
12032:   Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean ) : AnsiString
12033:   Function ALUTF8XMLTextElementDecode( const Src : AnsiString ) : AnsiString
12034:   Function ALUTF8HTMLDecode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
useNumRef:bool):AnsiString);
12035:   Function ALUTF8HTMLDecode( const Src : AnsiString ) : AnsiString
12036:   Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean ) : AnsiString
12037:   Function ALUTF8JavascriptDecode( const Src : AnsiString ) : AnsiString
12038:   Procedure ALHideHTMLUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12039:   Procedure ALCompactHTMLTagParams( TagParams : TALStrings )
12040: end;
12041:
12042: procedure SIRегистrier_ALInternetMessageCommon(CL: TPSPascalCompiler);
12043: begin
12044:   SIRегистrier_TALEMailHeader(CL);
12045:   SIRегистrier_TALNewsArticleHeader(CL);
12046:   Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
decodeRealName:Bool):AnsiString;
12047:   Function AlExtractEmailAddress( FriendlyEmail : AnsiString ) : AnsiString
12048:   Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString ) : AnsiString
12049:   Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString ) : AnsiString
12050:   Function AlGenerateInternetMessageID : AnsiString;
12051:   Function AlGenerateInternetMessageID1( ahostname : AnsiString ) : AnsiString;
12052:   Function ALDecodeQuotedPrintableString( src : AnsiString ) : AnsiString
12053:   Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12054: end;
12055:
12056: (*-----*)
12057: procedure SIRегистrier_ALFcnWinSock(CL: TPSPascalCompiler);
12058: begin
12059:   Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12060:   Function ALIPAddrToName( IPAddr : AnsiString ) : AnsiString
12061:   Function ALgetLocalIPs : TALStrings
12062:   Function ALgetLocalHostName : AnsiString
12063: end;
12064:
12065: procedure SIRегистrier_ALFcnCGI(CL: TPSPascalCompiler);

```

```

12066: begin
12067:   Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest : TALWebRequest; ServerVariables:TALStrings);
12068:   Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables : TALStrings; ScriptName , ScriptFileName : AnsiString; Url : AnsiString);
12069:   Procedure AlCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12070:   Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
12071:   ScriptFileName:AnsiString;Url:Ansistr;
12072:   Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
12073:   ScriptName , ScriptFileName : AnsiString; Url : AnsiString);
12074:   Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
12075:   : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12076:   Procedure AlCGIExec1(ScriptName,ScriptFileName , Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
12077:   WebRequest : TALIsapiRequest;
12078:   overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;
12079:   +'overloadedRequestContentStream:Tstream;var
12080:   ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12081:   Procedure AlCGIExec2(ScriptName,ScriptFileName , Url,X_REWRITE_URL,
12082:   InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
12083:   ResponseHeader : TALHTTPResponseHeader);
12084: end;
12077:
12078: procedure SIRegister_ALFcns(CL: TPSPascalCompiler);
12079: begin
12080:   TStartupInfoA' , 'TStartupInfo
12081:   'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12082:   SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12083:   SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12084:   SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12085:   SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12086:   SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12087:   SE_TCB_NAME','String 'SeTcbPrivilege
12088:   SE_SECURITY_NAME','String 'SeSecurityPrivilege
12089:   SE TAKE OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12090:   SE LOAD DRIVER_NAME','String 'SeLoadDriverPrivilege
12091:   SE SYSTEM PROFILE_NAME','String 'SeSystemProfilePrivilege
12092:   SE SYSTEMTIME NAME','String 'SeSystemtimePrivilege
12093:   SE PROF SINGLE PROCESS NAME','String 'SeProfileSingleProcessPrivilege
12094:   SE INC BASE PRIORITY NAME','String 'SeIncreaseBasePriorityPrivilege
12095:   SE CREATE PAGEFILE NAME','String 'SeCreatePagefilePrivilege
12096:   SE CREATE PERMANENT NAME','String 'SeCreatePermanentPrivilege
12097:   SE BACKUP NAME','String 'SeBackupPrivilege
12098:   SE RESTORE NAME','String 'SeRestorePrivilege
12099:   SE SHUTDOWN NAME','String 'SeShutdownPrivilege
12100:   SE DEBUG NAME','String 'SeDebugPrivilege
12101:   SE AUDIT NAME','String 'SeAuditPrivilege
12102:   SE SYSTEM ENVIRONMENT NAME','String 'SeSystemEnvironmentPrivilege
12103:   SE CHANGE NOTIFY NAME','string 'SeChangeNotifyPrivilege
12104:   SE REMOTE SHUTDOWN NAME','String 'SeRemoteShutdownPrivilege
12105:   SE UNDOCK NAME','String 'SeUndockPrivilege
12106:   SE SYNC AGENT NAME','String 'SeSyncAgentPrivilege
12107:   SE ENABLE DELEGATION NAME','String 'SeEnableDelegationPrivilege
12108:   SE MANAGE VOLUME NAME','String 'SeManageVolumePrivilege
12109:   Function AlGetEnvironmentString : AnsiString
12110:   Function ALWinExec32(const FileName,CurrentDir,
12111:   Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12112:   Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12113:   Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12114:   Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12115:   Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12116: end;
12117: procedure SIRegister_ALFcns(CL: TPSPascalCompiler);
12118: begin
12119:   Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
12120:   RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12121:   Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
12122:   RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12123:   Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
12124:   FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12125:   Function ALGetModuleName : ansistring
12126:   Function ALGetModuleFileNameWithoutExtension : ansistring
12127:   Function ALGetModulePath : ansiString
12128:   Function AlGetFileSize( const AFileName : ansistring) : int64
12129:   Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12130:   Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12131:   Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12132:   Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12133:   Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12134:   Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12135:   Function ALFileExists( const Path : ansiString) : boolean
12136:   Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12137:   Function ALCreateDir( const Dir : Ansistring) : Boolean
12138:   Function ALRemoveDir( const Dir : Ansistring) : Boolean
12139:   Function ALDeleteFile( const FileName : Ansistring) : Boolean
12140:   Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12141: end;
12142: procedure SIRegister_ALFcnsMime(CL: TPSPascalCompiler);

```

```

12141: begin
12142:   NativeInt', 'Integer
12143:   NativeUInt', 'Cardinal
12144:   Function ALMimeBase64EncodeString( const S : AnsiString ) : AnsiString
12145:   Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString ) : AnsiString
12146:   Function ALMimeBase64DecodeString( const S : AnsiString ) : AnsiString
12147:   Function ALMimeBase64EncodedSize( const InputSize : NativeInt ) : NativeInt
12148:   Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt ) : NativeInt
12149:   Function ALMimeBase64DecodedSize( const InputSize : NativeInt ) : NativeInt
12150:   Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12151:   Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12152:   Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12153:   Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt ) : NativeInt;
12154:   Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; '
12155:   + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12156:   Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12157:   Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
OutputBuf:TByteDynArray);
12158:   Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12159:   Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12160:   Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray):NativeInt;
12161:   Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer : TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12162:   Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
ByteBufferSpace:Cardinal):NativeInt;
12163:   Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12164:   Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12165:   Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12166:   Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12167:   Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12168:   Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12169:   'cALMimeBase64_ENCODED_LINE_BREAK', 'LongInt'( 76 );
12170:   'cALMimeBase64_DECODED_LINE_BREAK', 'LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3 );
12171:   'cALMimeBase64_BUFFER_SIZE', 'LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4 );
12172:   Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12173:   Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12174:   Function ALGetDefaultFileExtFromMimeType( aContentType : AnsiString ) : AnsiString
12175:   Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString ) : AnsiString
12176: end;
12177:
12178: procedure SIRegister_ALXmlDoc(CL: TPPascalCompiler);
12179: begin
12180:   'cALXMLNodeMaxListSize','LongInt'( Maxint div 16 );
12181:   FindClass( 'TOBJECT' ),'TALXMLNode
12182:   FindClass( 'TOBJECT' ),'TALXMLNodeList
12183:   FindClass( 'TOBJECT' ),'TALXMLDocument
12184:   TALXMLParseProcessingInstructionEvent', 'Procedure ( Sender:TObject; const Target,Data:AnsiString )
12185:   TALXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString )
12186:   TALXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12187:   +'nst Name : AnsiString; const Attributes : TALStrings )
12188:   TALXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString )
12189:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12190:   +'ntCDATA, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12191:   +'ntDocType, ntDocFragment, ntNotation )
12192:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12193:   TALXMLDocOptions', 'set of TALXMLDocOption
12194:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12195:   TALXMLParseOptions', 'set of TALXMLParseOption
12196:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12197:   PALPointerXMLNodeList', '^PALPointerXMLNodeList // will not work
12198:   SIRegister_EALXMLDocError(CL);
12199:   SIRegister_TALXMLNodeList(CL);
12200:   SIRegister_TALXMLNode(CL);
12201:   SIRegister_TALXmlElementNode(CL);
12202:   SIRegister_TALXmlAttributeNode(CL);
12203:   SIRegister_TALXmlTextNode(CL);
12204:   SIRegister_TALXmlDocumentNode(CL);
12205:   SIRegister_TALXmlCommentNode(CL);
12206:   SIRegister_TALXmlProcessingInstrNode(CL);
12207:   SIRegister_TALXmlCDataNode(CL);
12208:   SIRegister_TALXmlEntityRefNode(CL);
12209:   SIRegister_TALXmlEntityNode(CL);
12210:   SIRegister_TALXmlDocTypeNode(CL);
12211:   SIRegister_TALXmlDocFragmentNode(CL);
12212:   SIRegister_TALXmlNotationNode(CL);
12213:   SIRegister_TALXMLDocument(CL);
12214:   cALXMLUTF8EncodingStr', 'String 'UTF-8
12215:   cALxmlUTF8HeaderStr', 'String <?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>' +#13#10);
12216:   CALNSDelim,'String ':'
12217:   CALXML','String 'xml

```

```

12218: CALVersion', 'String 'version
12219: CALEncoding', 'String 'encoding
12220: CALStandalone', 'String 'standalone
12221: CALDefaultNodeIndent', 'String '
12222: CALXmlDocument', 'String 'DOCUMENT
12223: Function ALCreateEmptyXMLDocument( const Rootname : AnsiString ) : TalXMLDocument
12224: Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
EncodingStr:AnsiString);
12225: Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildnodeName,
ChildnodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12226: Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
ChildnodeName, ChildnodeValue : AnsiString; const Recurse : Boolean ) : TalxmlNode
12227: Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
Recurse: Boolean):TalxmlNode
12228: Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
AttributeName,AttributeValue : AnsiString; const Recurse : Boolean ) : TalxmlNode
12229: Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString ) :
AnsiString
12230: end;
12231:
12232: procedure SIRegister_TeCanvas(CL: TPSPPascalCompiler);
12233: //based on TEEProc, TeCanvas, TEEEngine, TChart
12234: begin
12235:   'TeePiStep','Double').setExtended( Pi / 180.0);
12236:   'TeeDefaultPerspective','LongInt'( 100 );
12237:   'TeeMinAngle','LongInt'( 270 );
12238:   'teeclMoneyGreen','LongWord').SetUInt( TColor( $C0DCC0 ) );
12239:   'teeclSkyBlue','LongWord').SetUInt( TColor( $F0CAA6 ) );
12240:   'teeclCream','LongWord( TColor( $F0FBFF ) );
12241:   'teeclMedGray','LongWord').SetUInt( TColor( $A4A0A0 ) );
12242:   'teeclMoneyGreen','LongWord').SetUInt( TColor( $C0DCC0 ) );
12243:   'teeclSkyBlue','LongWord').SetUInt( TColor( $F0CAA6 ) );
12244:   'teeclCream','LongWord').SetUInt( TColor( $F0FBFF ) );
12245:   'teeclMedGray','LongWord').SetUInt( TColor( $A4A0A0 ) );
12246:   'TA_LEFT','LongInt'( 0 );
12247:   'TA_RIGHT','LongInt'( 2 );
12248:   'TA_CENTER','LongInt'( 6 );
12249:   'TA_TOP','LongInt'( 0 );
12250:   'TA_BOTTOM','LongInt'( 8 );
12251:   'teePATCOPY','LongInt'( 0 );
12252:   'NumCirclePoints','Longint'( 64 );
12253:   'teeDEFAULT_CHARSET','LongInt'( 1 );
12254:   'teeANTIALIASED_QUALITY','LongInt'( 4 );
12255:   'TA_LEFT','LongInt'( 0 );
12256:   'bs_Solid','LongInt'( 0 );
12257:   'teepf24Bit','LongInt'( 0 );
12258:   'teepfDevice ','LongInt'( 1 );
12259:   'CM_MOUSELEAVE','LongInt'( 10000 );
12260:   'CM_SYSCOLORCHANGE','LongInt'( 10001 );
12261:   'DC_BRUSH','LongInt'( 18 );
12262:   'DC_PEN','LongInt'( 19 );
12263:   teeCOLORREF', 'LongWord
12264:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12265: //TNotifyEvent', 'Procedure ( Sender : TObject )
12266: SIRegister_TFilterRegion(CL);
12267: SIRegister_IFormCreator(CL);
12268: SIRegister_TTeeFilter(CL);
12269: //TFilterClass', 'class of TTeeFilter
12270: SIRegister_TFilterItems(CL);
12271: SIRegister_TConvolveFilter(CL);
12272: SIRegister_TBlurFilter(CL);
12273: SIRegister_TTeePicture(CL);
12274: TPenEndStyle', '( esRound, esSquare, esFlat )
12275: SIRegister_TChartPen(CL);
12276: SIRegister_TChartHiddenPen(CL);
12277: SIRegister_TDottedGrayPen(CL);
12278: SIRegister_TDDarkGrayPen(CL);
12279: SIRegister_TWhitePen(CL);
12280: SIRegister_TChartBrush(CL);
12281: TTeeView3DSScrolled', 'Procedure ( IsHoriz : Boolean)
12282: TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12283: SIRegister_TVview3DOptions(CL);
12284: FindClass('TOBJECT'),TTeeCanvas
12285: TTeeTransparency', 'Integer
12286: SIRegister_TTeeBlend(CL);
12287: FindClass('TOBJECT'),TCanvas3D
12288: SIRegister_TTeeShadow(CL);
12289: teeTGradientDirection', '(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12290: FindClass('TOBJECT'),TSubGradient
12291: SIRegister_TCustomTeeGradient(CL);
12292: SIRegister_TSubGradient(CL);
12293: SIRegister_TTeeGradient(CL);
12294: SIRegister_TTeeFontGradient(CL);
12295: SIRegister_TTeeFont(CL);
12296: TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12297: TCanvasTextAlign', 'Integer
12298: TTeeCanvasHandle', 'HDC
12299: SIRegister_TTeeCanvas(CL);

```

```

12300: TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12301: SIRegister_TFloatXYZ(CL);
12302: TPoint3D', 'record x : integer; y : integer; z : Integer; end
12303: TRGB', 'record blue: byte; green: byte; red: byte; end
12304: {TRGB=packed record
12305:   Blue : Byte;
12306:   Green : Byte;
12307:   Red : Byte;
12308: //$$IFDEF CLX //Alpha : Byte; // Linux end;}
12309:
12310: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 :
12311:   +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12312: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12313: TCanvas3DPlane', '( cpX, cpY, cpZ )
12314: //IInterface', 'IUnknown
12315: SIRegister_TCanvas3D(CL);
12316: SIRegister_TTeeCanvas3D(CL);
12317: TTrianglePoints', 'Array[0..2] of TPoint;
12318: TFourPoints', 'Array[0..3] of TPoint;
12319: Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12320: Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12321: Function Point3D( const x, y, z : Integer) : TPoint3D
12322: Procedure SwapDouble( var a, b : Double)
12323: Procedure SwapInteger( var a, b : Integer)
12324: Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12325: Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12326: Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12327: Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12328: Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12329: Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12330: Procedure UnClipCanvas( ACanvas : TCanvas)
12331: Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12332: Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12333: Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12334: 'TeeCharForHeight','String 'W
12335: 'DarkerColorQuantity','Byte').SetUInt( 128);
12336: 'DarkColorQuantity','Byte').SetUInt( 64);
12337: TButtonGetColorProc', 'Function : TColor
12338: SIRegister_TTeeButton(CL);
12339: SIRegister_TButtonColor(CL);
12340: SIRegister_TComboFlat(CL);
12341: Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12342: Function TeePoint( const ax, ay : Integer) : TPoint
12343: Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12344: Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12345: Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12346: Function OrientRectangle( const R : TRect) : TRect
12347: Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12348: Function PolygonBounds( const P : array of TPoint) : TRect
12349: Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12350: Function RGBValue( const Color : TColor) : TRGB
12351: Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12352: Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12353: Function PointAtDistance( AFrom, ATo : TPoint; Adist : Integer) : TPoint
12354: Function TeeCull( const P : TFourPoints) : Boolean;
12355: Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12356: TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12357: Procedure SmoothStretch( Src, Dst : TBitmap);
12358: Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12359: Function TeeDistance( const x, y : Double) : Double
12360: Function TeeLoadLibrary( const FileName : String) : HInst
12361: Procedure TeeFreeLibrary( hLibModule : HMODULE)
12362: Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12363: //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12364: Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
  Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12365:   SIRegister_ICanvasHyperlinks(CL);
12366:   SIRegister_ICanvasToolTips(CL);
12367:   Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12368: end;
12369:
12370: procedure SIRegister_ovcmisc(CL: TPPSPascalCompiler);
12371: begin
12372:   TOvcHdc', 'Integer
12373:   TOvcHWND', 'Cardinal
12374:   TOvcHdc', 'HDC
12375:   TOvcHWND', 'HWND
12376:   Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12377:   Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12378:   Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12379:   Function DefaultEpoch : Integer
12380:   Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12381:   Procedure FixRealPrim( P : PChar; DC : Char)
12382:   Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12383:   Function GetLeftButton : Byte
12384:   Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12385:   Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12386:   Function GetShiftFlags : Byte
12387:   Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont

```

```

12388: Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12389: Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet ) : string
12390: Function ovIsForegroundTask : Boolean
12391: Function ovTrimLeft( const S : string ) : string
12392: Function ovTrimRight( const S : string ) : string
12393: Function ovQuotedStr( const S : string ) : string
12394: Function ovWordCount( const S : string; const WordDelims : TCharSet ) : Integer
12395: Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12396: Function PtrDiff( const P1, P2 : PChar ) : Word
12397: Procedure PtrInc( var P, Delta : Word )
12398: Procedure PtrDec( var P, Delta : Word )
12399: Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer )
12400: Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY, SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer )
12401: Function ovMinI( X, Y : Integer ) : Integer
12402: Function ovMaxI( X, Y : Integer ) : Integer
12403: Function ovMinL( X, Y : LongInt ) : LongInt
12404: Function ovMaxL( X, Y : LongInt ) : LongInt
12405: Function GenerateComponentName( PF : TWInControl; const Root : string ) : string
12406: Function PartialCompare( const S1, S2 : string ) : Boolean
12407: Function PathEllipsis( const S : string; MaxWidth : Integer ) : string
12408: Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor ) : TBitmap
12409: Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas )
12410: Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap; TransparentColor : TColor )
12411: Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;TransparentColor : TColorRef)
12412: Function ovWidthOf( const R : TRect ) : Integer
12413: Function ovHeightOf( const R : TRect ) : Integer
12414: Procedure ovDebugOutput( const S : string )
12415: Function GetArrowWidth( Width, Height : Integer ) : Integer
12416: Procedure StripCharSeq( CharSeq : string; var Str : string )
12417: Procedure StripCharFromEnd( aChr : Char; var Str : string )
12418: Procedure StripCharFromFront( aChr : Char; var Str : string )
12419: Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12420: Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12421: Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12422: Function CreateEllipticRgn( p1, p2, p3, p4 : Integer ) : HRGN
12423: Function CreateEllipticRgnIndirect( const p1 : TRect ) : HRGN
12424: Function CreateFontIndirect( const p1 : TLogFont ) : HFONT
12425: Function CreateMetaFile( p1 : PChar ) : HDC
12426: Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12427: Function DrawText(hdc: HDC;lpString: PChar;nCount: Integer; var lpRect : TRect;uFormat:UINT):Integer
12428: Function DrawTextS(hdc: HDC;lpString:string;nCount: Integer; var lpRect: TRect;uFormat:UINT):Integer
12429: Function SetMapperFlags( DC : HDC; Flag : DWORD ) : DWORD
12430: Function SetGraphicsMode( hdc : HDC; iMode : Integer ) : Integer
12431: Function SetMapMode( DC : HDC; p2 : Integer ) : Integer
12432: Function SetMetaFileBitsEx( Size : UINT; const Data : PChar ) : HMETAFILE
12433: //Function SetPaletteEntries(Palette:HPalette;startIndex,NumEntries:UINT;var PaletteEntries):UINT
12434: Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF ) : COLORREF
12435: Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF ) : BOOL
12436: //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor ) : BOOL
12437: Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer ) : Integer
12438: Function StretchBlt(DestDC: HDC; X, Y, Width, Height : Int; SrcDC: HDC; XSrc, YSrc, SrcWidth, SrcHeight: Int; Rop:DWORD):BOOL
12439: Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer ) : BOOL
12440: Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth, SrcHeight:Int/Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD ) : Integer
12441: Function SetROP2( DC : HDC; p2 : Integer ) : Integer
12442: Function SetStretchBltMode( DC : HDC; StretchMode : Integer ) : Integer
12443: Function SetSystemPaletteUse( DC : HDC; p2 : UINT ) : UINT
12444: Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer ) : Integer
12445: Function SetTextColor( DC : HDC; Color : COLORREF ) : COLORREF
12446: Function SetTextAlign( DC : HDC; Flags : UINT ) : UINT
12447: Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer ) : Integer
12448: Function UpdateColors( DC : HDC ) : BOOL
12449: Function GetViewportExtEx( DC : HDC; var Size : TSize ) : BOOL
12450: Function GetViewportOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12451: Function GetWindowExtEx( DC : HDC; var Size : TSize ) : BOOL
12452: Function GetWindowOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12453: Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer ) : Integer
12454: Function InvertRgn( DC : HDC; p2 : HRGN ) : BOOL
12455: Function MaskBlt(DestDC: HDC; XDest, YDest, Width, Height : Integer; SrcDC : HDC; XScr, YScr : Integer; Mask : HBITMAP; xMask, yMask : Integer; Rop : DWORD ) : BOOL
12456: Function PlgBlt(DestDC: HDC; const PtsArray, SrcDC: HDC; XSrc, YSrc, Widt, Heigh: Int; Mask: HBITMAP; xMask, yMask: Int; Rop: DWORD):BOOL
12457: Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer ) : Integer
12458: Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer ) : Integer
12459: Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD ) : BOOL
12460: Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer ) : BOOL
12461: Function PlayMetaFile( DC : HDC; MF : HMETAFILE ) : BOOL
12462: Function PaintRgn( DC : HDC; RGN : HRGN ) : BOOL
12463: Function PtInRegion( RGN : HRGN; X, Y : Integer ) : BOOL
12464: Function PtVisible( DC : HDC; X, Y : Integer ) : BOOL
12465: Function RectInRegion( RGN : HRGN; const Rect : TRect ) : BOOL
12466: Function RectVisible( DC : HDC; const Rect : TRect ) : BOOL
12467: Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer ) : BOOL
12468: Function RestoreDC( DC : HDC; SavedDC : Integer ) : BOOL
12469: end;

```

```

12470:
12471: procedure SIRegister_ovcfiler(CL: TPSPPascalCompiler);
12472: begin
12473:   SIRegister_TOvcAbstractStore(CL);
12474:   //PEXPropInfo', '^TExPropInfo // will not work
12475:   //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12476:   SIRegister_TOvcPropertyList(CL);
12477:   SIRegister_TOvcDataFiler(CL);
12478:   Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12479:   Procedure UpdateStoredList( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12480:   Function CreateStoredItem( const CompName, PropName : string) : string
12481:   Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12482:   //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12483: end;
12484:
12485: procedure SIRegister_ovccoco(CL: TPSPPascalCompiler);
12486: begin
12487:   'ovsetsize','LongInt'( 16);
12488:   'etSyntax','LongInt'( 0);
12489:   'etSemantic','LongInt'( 1);
12490:   'chCR','Char #13);
12491:   'chLF','Char #10);
12492:   'chLineSeparator','chCR);
12493:   SIRegister_TCocoError(CL);
12494:   SIRegister_TCommentItem(CL);
12495:   SIRegister_TCommentList(CL);
12496:   SIRegister_TSymbolPosition(CL);
12497:   TGenListType', '( glNever, glAlways, glOnError )
12498:   TovBitSet', 'set of Integer
12499:   //PStartTable', '^TStartTable // will not work
12500:   'TovCharSet', 'set of AnsiChar
12501:   TAFTERGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12502:   TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12503:   TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12504:   TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12505:   TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12506:     +'osition; const Data : string; ErrorType : integer)
12507:   TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12508:   TGetCH', 'Function ( pos : longint) : char
12509:   TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12510:   SIRegister_TCocoRScanner(CL);
12511:   SIRegister_TCocoRGrammar(CL);
12512:   '_EF','Char #0);
12513:   '_TAB','Char').SetString( #09);
12514:   '_CR','Char').SetString( #13);
12515:   '_LF','Char').SetString( #10);
12516:   '_EL','','').SetString( _CR);
12517:   '_EOF','Char').SetString( #26);
12518:   'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12519:   'minErrDist','LongInt'( 2);
12520:   Function ovPadL( S : string; ch : char; L : integer) : string
12521: end;
12522:
12523: TFormatSettings = record
12524:   CurrencyFormat: Byte;
12525:   NegCurrFormat: Byte;
12526:   ThousandSeparator: Char;
12527:   DecimalSeparator: Char;
12528:   CurrencyDecimals: Byte;
12529:   DateSeparator: Char;
12530:   TimeSeparator: Char;
12531:   ListSeparator: Char;
12532:   CurrencyString: string;
12533:   ShortDateFormat: string;
12534:   LongDateFormat: string;
12535:   TimeAMString: string;
12536:   TimePMString: string;
12537:   ShortTimeFormat: string;
12538:   LongTimeFormat: string;
12539:   ShortMonthNames: array[1..12] of string;
12540:   LongMonthNames: array[1..12] of string;
12541:   ShortDayNames: array[1..7] of string;
12542:   LongDayNames: array[1..7] of string;
12543:   TwoDigitYearCenturyWindow: Word;
12544: end;
12545:
12546: procedure SIRegister_OvcFormatSettings(CL: TPSPPascalCompiler);
12547: begin
12548:   Function ovFormatSettings : TFormatSettings
12549: end;
12550:
12551: procedure SIRegister_ovcstr(CL: TPSPPascalCompiler);
12552: begin
12553:   TOvc CharSet', 'set of Char
12554:   ovBTable', 'array[0..255] of Byte
12555:   //BTable = array[0..{$IFDEF UNICODE}{$ENDIF}{$ELSE}{$ENDIF}{$ENDIF} of Byte;
12556:   Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12557:   Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar

```

```

12558: Function BinaryWPChar( Dest : PChar; W : Word ) : PChar
12559: Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12560: Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12561: Function BMSearchC(var Buffer,BufLength:Cardinal; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12562: Function CharStarPChar( Dest : PChar; C : Char; Len : Cardinal ) : PChar
12563: Function DatabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12564: Function HexBPChar( Dest : PChar; B : Byte) : PChar
12565: Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12566: Function HexPtrPChar( Dest : PChar; P : TObject ) : PChar
12567: Function HexWPChar( Dest : PChar; W : Word) : PChar
12568: Function LoCaseChar( C : Char ) : Char
12569: Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12570: Function StrChDeletePrim( P : PChar; Pos : Cardinal ) : PChar
12571: Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal ) : PChar
12572: Function StrChPos( P : PChar; C : Char; var Pos : Cardinal ) : Boolean
12573: Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12574: Function StrSCopy( Dest, S : PChar; Pos, Count : Cardinal ) : PChar
12575: Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal ) : PChar
12576: Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal ) : PChar
12577: Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal ) : PChar
12578: Function StrStPos( P, S : PChar; var Pos : Cardinal ) : Boolean
12579: Function StrToIntLongPChar( S : PChar; var I : LongInt ) : Boolean
12580: Procedure TrimAllSpacesPChar( P : PChar )
12581: Function TrimEmbeddedZeros( const S : string ) : string
12582: Procedure TrimEmbeddedZerosPChar( P : PChar )
12583: Function TrimTrailPrimPChar( S : PChar ) : PChar
12584: Function TrimTrailPChar( Dest, S : PChar ) : PChar
12585: Function TrimTrailingZeros( const S : string ) : string
12586: Procedure TrimTrailingZerosPChar( P : PChar )
12587: Function UpCaseChar( C : Char ) : Char
12588: Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet ) : Boolean
12589: Function ovc32StringIsCurrentCodePage( const S : WideString ) : Boolean;
12590: //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal ) : Boolean;
12591: end;
12592:
12593: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12594: begin
12595:   //PraiseFrame', '^TRaiseFrame // will not work
12596:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : __Poin'
12597:   +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12598:   Procedure SafeCloseHandle( var Handle : THandle )
12599:   Procedure ExchangeInteger( X1, X2 : Integer )
12600:   Procedure FillInteger( const Buffer, Size, Value : Integer )
12601:   Function LongMulDiv( Multi, Mult2, Div1 : Longint ) : Longint
12602:   Function afCompareMem( P1, P2 : TObject; Length : Integer ) : Boolean
12603:
12604:   FILENAME_ADVAPI32      = 'ADVAPI32.DLL';
12605:   function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12606:   function AbortSystemShutdown(lpMachineName: PKOLOChar): BOOL; stdcall;
12607:   function AccessCheckAndAuditAlarm(SubsystemName: PKOLOChar;
12608:     HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12609:     SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12610:     const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12611:     var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12612:   function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLOChar;
12613:     HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12614:     SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12615:     AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12616:     ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12617:     var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12618:   function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLOChar;
12619:     HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12620:     SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12621:     AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12622:     ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12623:     var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12624:   function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLOChar): BOOL; stdcall;
12625:   function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLOChar): BOOL; stdcall;
12626:   function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLOChar;
12627:     lpCommandLine: PKOLOChar; lpProcessAttributes: PSecurityAttributes;
12628:     lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12629:     dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLOChar;
12630:     const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12631:   function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12632:   function GetFileSecurity(lpFileName: PKOLOChar; RequestedInformation: SECURITY_INFORMATION;
12633:     pSecurityDescriptor: PSecurityDescriptor; nLength:DWORD; var lpnLengthNeeded: DWORD):BOOL; stdcall;
12634:   function GetUserName(lpBuffer: PKOLOChar; var nSize: DWORD): BOOL; stdcall;
12635:   function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLOChar;
12636:     dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12637:   function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLOChar;
12638:     dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12639:   function LookupAccountName(lpSystemName, lpAccountName: PKOLOChar;
12640:     Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLOChar;
12641:     var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12642:   function LookupAccountSid(lpSystemName: PKOLOChar; Sid: PSID;
12643:     Name: PKOLOChar; var cbName: DWORD; ReferencedDomainName: PKOLOChar;
12644:     var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12645:   function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLOChar;
12646:     lpDisplayName: PKOLOChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;

```

```

12647:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12648:         var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12649:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12650:         var lpLuid: TLargeInteger): BOOL; stdcall;
12651:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12652:         HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12653:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12654:         HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12655:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12656:         ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12657:         ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12658:         var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12659:         var GenerateOnClose: BOOL): BOOL; stdcall;
12660:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12661:         HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12662:         var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12663:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12664:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12665:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12666:         ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12667:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12668:         lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12669:         var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12670:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12671:         var phkResult: HKEY): Longint; stdcall;
12672:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12673:         var phkResult: HKEY): Longint; stdcall;
12674:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12675:         Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12676:         lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12677:         lpdwDisposition: PDWORD): Longint; stdcall;
12678:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12679:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12680:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12681:         var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12682:         lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12683:     function RegEnumKey(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar; cbName: DWORD): Longint; stdcall;
12684:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12685:         var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12686:         lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12687:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12688:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar; var phkResult: HKEY): Longint; stdcall;
12689:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12690:         ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12691:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12692:         lpcbClass: PDWORD; lpReserved: Pointer;
12693:         lpcbSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12694:         lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12695:         lpftLastWriteTime: PFileTime): Longint; stdcall;
12696:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12697:         NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotSize: DWORD): Longint; stdcall;
12698:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12699:         lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12700:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12701:         lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12702:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12703:         lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12704:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12705:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12706:         lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12707:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12708:         dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12709:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12710:         Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12711:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12712:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12713:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12714:         dwEventID: DWORD; lpUserId: Pointer; wNumStrings: Word;
12715:         dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12716:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12717:         pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12718:
12719:     Function wAddAtom( lpString : PKOLChar ) : ATOM
12720:     Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL ) : THandle
12721:         //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
12722:         lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD ) : BOOL
12723:         //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig ) : BOOL
12724:         Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
12725:             lpString2 : PKOLChar; cchCount2 : Integer ) : Integer
12726:         Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL ) : BOOL
12727:             //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
12728:                 TFnProgressRoutine; lpData : Pointer; pbcancel : PBool; dwCopyFlags : DWORD ) : BOOL
12729:         Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes ) : BOOL
12730:         Function wCreateDirectoryEx(lpTemplateDirectory,
12731:             lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribs):BOOL;
12732:         Function wCreateEvent(lpEventName:PSecurityAttribs:PSecurityAttrib:bManualReset,
12733:             bInitialState:BOOL;lpName:PKOLChar):THandle;
12734:         Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
12735:             PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD; hTemplateFile:THandle ) : THandle

```

```

12730: Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar ) : THandle
12731: Function wCreateHardLink( lpFileName,
lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12732: Function CreateMailslot( lpName : PKOLChar; MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes ):THandle;
12733: Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes ) : THandle
12734: //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
lpProcessInfo:TProcessInformation):BOOL
12735: Function wCreateSemaphore( lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
Longint; lpName : PKOLChar ) : THandle
12736: Function wCreateWaitableTimer( lpTimerAttrbs:PSecurityAttrbs;bManualReset:BOOL;lpTimerName:PKOLChar ):THandle;
12737: Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar ) : BOOL
12738: Function wDeleteFile( lpFileName : PKOLChar ) : BOOL
12739: Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL ) : BOOL
12740: //Function
wEnumCalendarInfo( lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE ):BOOL;
12741: //Function wEnumDateFormats( lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD ) : BOOL
12742: //Function
wEnumResourceNames( hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lpParam:Longint):BOOL;
12743: //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lpParam:Longint):BOOL;
12744: //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12745: //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12746: //Function wEnumTimeFormats( lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD ):BOOL;
12747: Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD ) : DWORD
12748: Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar )
12749: //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12750: Function wFindAtom( lpString : PKOLChar ) : ATOM
12751: Function wFindFirstChangeNotification( lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD ):THandle;
12752: Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData ) : THandle
12753: //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindIndexInfoLevels; lpFindFileData :
Pointer; fSearchOp : TFindIndexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD ) : BOOL
12754: Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData ) : BOOL
12755: Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar ) : HRSRC
12756: Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word ) : HRSRC
12757: Function wFoldString( dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer ):Integer;
12758: //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer ) : DWORD
12759: Function wFreeEnvironmentStrings( EnvBlock : PKOLChar ) : BOOL
12760: Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer ) : UINT
12761: Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD ) : BOOL
12762: Function wGetCommandLine : PKOLChar
12763: //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD ) : DWORD
12764: Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD ) : BOOL
12765: Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : WORD ) : WORD
12766: //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer ) : Integer
12767: Function wGetCurrentDirectory( nBufLength : DWORD; lpBuffer : PKOLChar ) : DWORD
12768: //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSysteMTime; lpFormat : PKOLChar;
lpDateStr : PKOLChar; cchDate : Integer ) : Integer
12769: //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD ):BOOL
12770: Function wGetFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD ) : BOOL
12771: //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger ) : BOOL
12772: Function wGetDriveType( lpRootPathName : PKOLChar ) : UINT
12773: Function wGetEnvironmentStrings : PKOLChar
12774: Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD ) : DWORD;
12775: Function wGetFileAttributes( lpFileName : PKOLChar ) : DWORD
12776: //Function
wGetFileAttributesEx( lpFileName:PKOLChar,fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12777: Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12778: //Function wGetLocaleInfo(Locale:LCID; LCTYPE:LCTYPE;lpLCDData:PKOLChar;cchData:Integer): Integer
12779: Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar ) : DWORD
12780: Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD ) : DWORD
12781: Function wGetModuleHandle( lpModuleName : PKOLChar ) : HMODULE
12782: //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD ) : BOOL
12783: //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
lpNumberStr : PKOLChar; cchNumber : Integer ) : Integer
12784: Function wGetPrivateProfileInt( lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12785: Function wGetPrivateProfileSection(lpAppName:PKOLChar;lpRtrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12786: Function wGetPrivateProfileSectionNames(lpszReturnBuffe:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12787: Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr : PKOLChar;
nSize:DWORD; lpFileName : PKOLChar ) : DWORD
12788: Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer ) : UINT
12789: Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD ) : DWORD
12790: Function wGetProfileString(lpAppName,lpKeyName,
lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12791: Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD ) : DWORD

```

```

12792: //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12793: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
12794: lpCharType):BOOL
12794: Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT ) : UINT
12795: Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar ):UINT
12796: Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar ) : DWORD
12797: //Function
12798: wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12799: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo ) : BOOL
12800: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
12801: : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
12802: lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD ) : BOOL
12803: Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT ) : UINT
12804: Function wGlobalAddAtom( lpString : PKOLChar ) : ATOM
12805: Function wGlobalFindAtom( lpString : PKOLChar ) : ATOM
12806: Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer ) : UINT
12807: Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT ) : BOOL
12808: Function wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int
12809: Function wLoadLibrary( lpLibFileName : PKOLChar ) : HMODULE
12810: Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD ) : HMODULE
12811: Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar ) : BOOL
12812: Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD ) : BOOL
12813: //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
12814: TFnProgressRoutine; lpData : Pointer; dwFlags : DWORD ) : BOOL
12815: Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar ) : THandle
12816: Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar ):THandle
12817: Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar ) : THandle
12818: Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar ):THandle
12819: Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12820: Procedure wOutputDebugString( lpOutputString : PKOLChar )
12821: //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
12822: lpNumberOfEventsRead:DWORD):BOOL;
12823: Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD ) : DWORD
12824: //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12825: //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
12826: lpNumberOfCharsRead : DWORD; lpReserved : Pointer ) : BOOL
12827: //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
12828: lpNumOfEventsRead:DWORD):BOOL;
12829: //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
12830: : TCoord; var lpReadRegion : TSmallRect ) : BOOL
12831: //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
12832: DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD ) : BOOL
12833: Function wRemoveDirectory( lpPathName : PKOLChar ) : BOOL
12834: //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
12835: lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo ) : BOOL
12836: Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
12837: lpFilePart:PKOLChar):DWORD;
12838: Function wSetComputerName( lpComputerName : PKOLChar ) : BOOL
12839: Function wSetConsoleTitle( lpConsoleTitle : PKOLChar ) : BOOL
12840: Function wSetCurrentDirectory( lpPathName : PKOLChar ) : BOOL
12841: //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCCommConfig; dwSize : DWORD ) : BOOL
12842: Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar ) : BOOL
12843: Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD ) : BOOL
12844: //Function wSetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCDat : PKOLChar ) : BOOL
12845: Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar ) : BOOL
12846: //Function wUpdateResource(hUpdate:THandle;lpType,
12847: lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12848: Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD ) : DWORD
12849: Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD ) : BOOL
12850: //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
12851: DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer ) : BOOL
12852: //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
12853: var lpNumberOfEventsWritten : DWORD ) : BOOL
12854: //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
12855: TCoord; var lpWriteRegion : TSmallRect ) : BOOL
12856: //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
12857: DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12858: Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar ) : BOOL
12859: Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar ) : BOOL
12860: Function wWriteProfileSection( lpAppName, lpString : PKOLChar ) : BOOL
12861: Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar ) : BOOL
12862: Function wlstrcat( lpString1, lpString2 : PKOLChar ) : PKOLChar
12863: Function wlstrcmp( lpString1, lpString2 : PKOLChar ) : Integer
12864: Function wlstrcmpi( lpString1, lpString2 : PKOLChar ) : Integer
12865: Function wlstrcpy( lpString1, lpString2 : PKOLChar ) : PKOLChar
12866: Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer ) : PKOLChar
12867: Function wlstrlen( lpString : PKOLChar ) : Integer
12868: //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruct :
12869: PNetConnectInfoStruct ) : DWORD
12870: //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
12871: lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12872: //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
12873: lpUserName:PKOLChar; dwFlags : DWORD ) : DWORD
12874: Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar ) : DWORD
12875: Function wWNetCancelConnection( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL ) : DWORD
12876: Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL ) : DWORD
12877: //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct ) : DWORD
12878: //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct ) : DWORD

```

```

12860: //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12861: Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12862: Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
: PKOLChar; nNameBufSize : DWORD) : DWORD
12863: //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12864: Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12865: //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12866: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
lpBufferSize:DWORD):DWORD;
12867: Function wWNet GetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12868: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12869: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer ) : DWORD
12870: //Function wWNetUseConnection(hwndOwner:HWND;var
lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12871: Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12872: Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12873: Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12874: Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12875: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12876: //Func wGetPrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12877: //Func wWritePrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12878: Function wAddFontResource(FileName : PKOLChar) : Integer
12879: //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12880: Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12881: Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12882: //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12883: //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12884: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12885: Function wCreateFont( nHeight, nWidth, nEscapement, nOrientation, fnWeight : Integer; fdwItalic,
fdwUnderline, fdwStrikeOut, fdwCharSet, fdwOutputPrec, fdwClipPrecision, fdwQualy,
fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT
12886: Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12887: //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12888: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12889: Function wCreateMetaFile( p1 : PKOLChar) : HDC
12890: Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12891: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
pPort:PKOLChar;iIdx:Int;out:PKOLChar;DevMod:PDeviceMode):Int;
12892: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFnFontEnumProc; p4 : LPARAM) : BOOL
12893: //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL;
12894: //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fnItemProc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12895: //Function wEnumICMPprofiles( DC : HDC; ICMPProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12896: //Function wExtTextOut(DC:HDC;X,
Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12897: //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12898: //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12899: //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12900: //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12901: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12902: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12903: Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12904: Function wGetEnhMetaFileDescription( pl : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12905: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12906: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:PGlyphMetrics; cbBuffer : DWORD;
lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12907: Function wGetICMPProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12908: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12909: Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12910: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12911: //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12912: //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12913: Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12914: Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12915: Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12916: //Function wGetTextMetrics( DC : HDC; var TM : TTTextMetric) : BOOL
12917: Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12918: Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12919: //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12920: //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12921: Function wSetICMPProfile( DC : HDC; Name : PKOLChar) : BOOL
12922: //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12923: Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12924: Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12925: Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12926: //Function wwglUseFontOutlines(p1:DC;p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12927: Function wAppendMenu( hMenu : HMENU; uFlags, uidNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12928: Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12929: //Function
wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12930: //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12931: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
dwFlags : DWORD; lParam : Pointer) : Longint
12932: Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12933: Function wCharLower( lpsz : PKOLChar) : PKOLChar

```

```

12934: Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12935: Function wCharNext( lpsz : PKOLChar ) : PKOLChar
12936: //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12937: Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar ) : PKOLChar
12938: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12939: Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12940: Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12941: Function wCharUpper( lpsz : PKOLChar ) : PKOLChar
12942: Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12943: Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer ) : Integer
12944: Function wCreateAcceleratorTable( var Accel, Count : Integer ) : HACCEL
12945: //Function wCreateDesktop(lpszDesktop,
12946: lpszDevice:PKOLChar,pDevMode:PDeviceMode,dwFlags:DWORD,dwDesiredAccess:DWORD;lpsa:PSecurityAttrs):HDESK
12947: //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
12948: HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND
12949: //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
12950: lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND
12951: Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
12952: hWndParent : HWND; hInstance : HINST; lParam : LPARAM ) : HWND
12953: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
12954: nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12955: //Function wCreateWindowStation(lpinsta:PKOLChar;dwReserv,
12956: dwDesiredAccess:DWORD;lpsa:PSecurityAttrs):HWINSTA;
12957: Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12958: Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12959: Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM ) : LRESULT
12960: Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam: LPARAM ) : LRESULT
12961: //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
12962: : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12963: //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
12964: : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12965: Function wDispatchMessage( const lpMsg : TMsg ) : Longint
12966: Function wDlgDirList( hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
nIDStaticPath:Integer;uFileType:UINT):Integer;
12967: Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
nIDStaticPath:Int;uFileType:UINT):Int;
12968: Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12969: Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer ) : BOOL
12970: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
12971: cy:Int;Flags:UINT):BOOL;
12972: Function wDrawText( hdc:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12973: Function wFindWindow( lpClassName, lpWindowName : PKOLChar ) : HWND
12974: Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar ) : HWND
12975: //Function wGetAltTabInfo( hwnd:HWND;iItem:Int;var
12976: pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12977: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass ) : BOOL
12978: //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx ) : BOOL
12979: Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer ) : Integer
12980: Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12981: Function wGetDlgItemText( hDlg : HWND;nIDDlgeItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12982: Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer ) : Integer
12983: Function wGetKeyboardLayoutName( pwszKLID : PKOLChar ) : BOOL
12984: //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo ) : BOOL
12985: Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12986: Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT ) : BOOL
12987: Function wGetProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12988: //Function wGetTabbedTextExtent( HDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
12989: lpnTabStopPositions ) : DWORD
12990: //Function wGetObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
12991: lpnLengthNeed:DWORD)BOOL;
12992: Function wGetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
12993: Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT ) : UINT
12994: Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer ) : Integer
12995: Function wGetWindowTextLength( hWnd : HWND ) : Integer
12996: //Function wGrayString(hdc:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
12997: nHeight:Int):BOOL;
12998: Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12999: //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
13000: Function wIsCharAlpha( ch : KOLChar ) : BOOL
13001: Function wIsCharAlphaNumeric( ch : KOLChar ) : BOOL
13002: Function wIsCharLower( ch : KOLChar ) : BOOL
13003: Function wIsCharUpper( ch : KOLChar ) : BOOL
13004: Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg ) : BOOL
13005: Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar ) : HACCEL
13006: Function wLoadBitmap( hInst : HINST; lpBitmapName : PKOLChar ) : HBITMAP
13007: Function wLoadCursor( hInst : HINST; lpCursorName : PKOLChar ) : HCURSOR
13008: Function wLoadCursorFromFile( lpFileName : PKOLChar ) : HCURSOR
13009: Function wLoadIcon( hInst : HINST; lpIconName : PKOLChar ) : HICON
13010: Function wLoadImage(hInst:HMENUP;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
13011: Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT ) : HKL
13012: Function wLoadMenu( hInst : HINST; lpMenuName : PKOLChar ) : HMENU
13013: //Function wLoadMenuItemDirect( lpMenuItemTemplate : Pointer ) : HMENU
13014: Function wLoadString(hInst:HMENUP;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
13015: Function wMapVirtualKey( uCode, uMapType : UINT ) : UINT
13016: Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL ) : UINT
13017: Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT ) : Integer
13018: Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
13019: //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams ) : BOOL

```

```

13008: Function wModifyMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
13009: //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR ) : BOOL
13010: //Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD ) : BOOL
13011: //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
13012: Function wOpenDesktop( lpszDesktop : PKOLChar; dwFlags:DWORD; fInherit:BOOL; dwDesiredAccess:DWORD ) : HDESK
13013: Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD ) : HWINSTA
13014: Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND; wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT ) : BOOL
13015: Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13016: Function wPostThreadMessage( idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13017: Function wPostThreadMessage( idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13018: Function wRealGetWindowClass( hWnd : HWND; pszType : PKOLChar; cchType : UINT ) : UINT
13019: // Function wRegisterClass( const lpWndClass : TWndClass ) : ATOM
13020: // Function wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM
13021: Function wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
13022: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13023: Function wRegisterWindowMessage( lpString : PKOLChar ) : UINT
13024: Function wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THandle
13025: Function wSendDlgItemMessage( hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13026: Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
13027: //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
lpResultCallBack : TFNSendAsyncProc; dwData : DWORD ) : BOOL
13028: Function wSendMessageTimeout( hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
lpdwResult:DWORD ) : LRESULT
13029: Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13030: Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
13031: Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar ) : BOOL
13032: //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
13033: Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle ) : BOOL
13034: // Function wSetUserObjectInformation(hObject:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
13035: Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint
13036: Function wSetWindowText( hWnd : HWND; lpString : PKOLChar ) : BOOL
13037: //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc ) : HHOOK
13038: //Function wSetWindowsHookEx(idHook:Integer,lpfn:TFNHookProc,hmod:HINST;dwThreadId:DWORD):HHOOK;
13039: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
13040: Function wTabbedTextOut( hdc:HDC;x,y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
lpnTabStopPositions,nTabOrigin:Int):Longint;
13041: Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg ) : Integer
13042: Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST ) : BOOL
13043: Function wVKeyScan( ch : KOLChar ) : SHORT
13044: Function wVKeyScanEx( ch : KOLChar; dwhkl : HKL ) : SHORT
13045: Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD ) : BOOL
13046: Function wwsprintf( Output : PKOLChar; Format : PKOLChar ) : Integer
13047: Function wwvssprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list ) : Integer
13048:
13049: //TestDrive!
13050: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL'
13051: 'PROC_CONVERTSINTOSTRINGSIDA','String').SetString( 'ConvertSintToStringSida'
13052: Function GetDomainUserSids( const domainName:String;const userName:String; var foundDomain:String):String;
13053: Function GetLocalUserSidStr( const UserName : string ) : string
13054: Function getPid4user( const domain : string; const user : string; var pid : dword ) : boolean
13055: Function Impersonate2User( const domain : string; const user : string ) : boolean
13056: Function GetProcessUserBpid( pid : DWORD; var UserName, Domain : AnsiString ) : Boolean
13057: Function KillProcessbyname( const exename : string; var found : integer ) : integer
13058: Function getWinProcessList : TStringList
13059: function WaitTilClose(hWnd: Integer): Integer;
13060: function DoUserMsgs: Boolean;
13061: function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13062: procedure ShowMsg(hParent: Integer; const Mess, Title: String); //modal but NOT blockable
13063: procedure DeleteMsgForm(Handle: Integer);
13064: procedure DisableForms;
13065: function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13066: end;
13067:
13068: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13069: begin
13070: 'AfMaxSyncSlots','LongInt'( 64 );
13071: 'AfSynchronizeTimeout','LongInt'( 2000 );
13072: TafSyncSlotID', 'DWORD
13073: TafSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end';
13074: TafSafeSyncEvent', 'Procedure ( ID : TafSyncSlotID )
13075: TafSafeDirectSyncEvent', 'Procedure
13076: Function AfNewSyncSlot( const AEvent : TafSafeSyncEvent ) : TafSyncSlotID
13077: Function AfReleaseSyncSlot( const ID : TafSyncSlotID ) : Boolean
13078: Function AfEnableSyncSlot( const ID : TafSyncSlotID; Enable : Boolean ) : Boolean
13079: Function AfValidateSyncSlot( const ID : TafSyncSlotID ) : Boolean
13080: Function AfSyncEvent( const ID : TafSyncSlotID; Timeout : DWORD ) : Boolean
13081: Function AfDirectSyncEvent( Event : TafSafeDirectSyncEvent; Timeout : DWORD ) : Boolean
13082: Function AfIsSyncMethod : Boolean
13083: Function AfSyncWnd : HWnd
13084: Function AfSyncStatistics : TafSyncStatistics
13085: Procedure AfClearSyncStatistics
13086: end;
13087:
13088: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13089: begin
13090: 'fBinary','LongWord')($00000001);
13091: 'fParity','LongWord')($00000002);
13092: 'fOutxCtsFlow','LongWord').SetUInt( $00000004 );
13093: 'fOutxDsrFlow','LongWord')($00000008 );

```

```

13094: 'fDtrControl', 'LongWord')($00000030);
13095: 'fDtrControlDisable', 'LongWord')($00000000);
13096: 'fDtrControlEnable', 'LongWord')($00000010);
13097: 'fDtrControlHandshake', 'LongWord')($00000020);
13098: 'fDsrsensitivity', 'LongWord')($00000040);
13099: 'ftxContinueOnKoff', 'LongWord')($00000080);
13100: 'fOutX', 'LongWord')($00000100);
13101: 'fInX', 'LongWord')($00000200);
13102: 'fErrorChar', 'LongWord')($00000400);
13103: 'fNull', 'LongWord')($00000800);
13104: 'fRtsControl', 'LongWord')($00003000);
13105: 'fRtsControlDisable', 'LongWord')($00000000);
13106: 'fRtsControlEnable', 'LongWord')($00001000);
13107: 'fRtsControlHandshake', 'LongWord')($00002000);
13108: 'fRtsControlToggle', 'LongWord')($00003000);
13109: 'fAbortOnError', 'LongWord')($00004000);
13110: 'fDummy2', 'LongWord')($FFFF8000);
13111: TAfcCoreEvent', '(ceOutFree, ceLineEvent, ceNeedReadData, ceException)
13112: FindClass('TObject'), 'EAfComPortCoreError
13113: FindClass('TObject'), 'TAfComPortCore
13114: TAfcComPortCoreEvent', 'Procedure (Sender : TAfcComPortCore; Event'
13115: +'tKind : TAfcCoreEvent; Data : DWORD)
13116: SIRegister_TAfcComPortCoreThread(CL);
13117: SIRegister_TAfcComPortEventThread(CL);
13118: SIRegister_TAfcComPortWriteThread(CL);
13119: SIRegister_TAfcComPortCore(CL);
13120: Function FormatDeviceName(PortNumber : Integer) : string
13121: end;
13122:
13123: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13124: begin
13125: TAfIOFileStreamEvent', 'Function (const fileName : String; mode: Word) : TStream
13126: TAfIOFileExistsEvent', 'Function (const fileName : String) : Boolean
13127: SIRegister_TApplicationFileIO(CL);
13128: TDataFileCapability', '(dfcRead, dfcWrite)
13129: TDataFileCapabilities', 'set of TDataFileCapability
13130: SIRegister_TDatafile(CL);
13131: //TDataFileClass', 'class of TDataFile
13132: Function ApplicationFileIODefined : Boolean
13133: Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone): TStream
13134: Function FilestreamExists(const fileName: String) : Boolean
13135: //Procedure Register
13136: end;
13137:
13138: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13139: begin
13140: TALFBXFieldType', '(uftUnknown, uftNumeric, uftChar, uftVarchar'
13141: +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecision'
13142: +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13143: TALFBXScale', 'Integer
13144: FindClass('TObject'), 'EALFBXConvertError
13145: SIRegister_EALFBXError(CL);
13146: SIRegister_EALFBXException(CL);
13147: FindClass('TObject'), 'EALFBXGFixError
13148: FindClass('TObject'), 'EALFBXDSQLError
13149: FindClass('TObject'), 'EALFBXDynError
13150: FindClass('TObject'), 'EALFBXBakError
13151: FindClass('TObject'), 'EALFBXGSecError
13152: FindClass('TObject'), 'EALFBXLicenseError
13153: FindClass('TObject'), 'EALFBXStatError
13154: //EALFBXExceptionClass', 'class of EALFBXError
13155: TALFBXCharacterSet', '(csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13156: +'37, csDOS850, csDOS852, csDOS853, csDOS860, csDOS861, csDOS863, csDOS865, '
13157: +'csEUUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13158: +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252, '
13159: +'csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13160: +'SDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13161: +'4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13162: +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13163: TALFBXTransParam', '(tpConsistency, tpConcurrency, tpShared, tp'
13164: +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13165: +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13166: +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13167: TALFBXTransParams', 'set of TALFBXTransParam
13168: Function ALFBXStrToCharacterSet(const CharacterSet : AnsiString) : TALFBXCharacterSet
13169: Function ALFBXCreateDBParams(Params : AnsiString; Delimiter : Char) : AnsiString
13170: Function ALFBXCreateBlobParams(Params : AnsiString; Delimiter : Char) : AnsiString
13171: 'CALFBXMaxParamLength', 'LongInt'(125);
13172: TALFBXParamsFlag', '(pfNotInitialized, pfNotNullable )
13173: TALFBXParamsFlags', 'set of TALFBXParamsFlag
13174: //PALFBXSQLVar', '^PALFBXSQLVar // will not work
13175: //PALFBXSQLDaData', '^PALFBXSQLDaData // will not work
13176: TALFBXStatementType', '(stSelect, stInsert, stUpdate, stDelete, '
13177: +'stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommis'
13178: +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13179: SIRegister_TALFBXSQLDA(CL);
13180: //PALFBXPtrArray', '^PALFBXPtrArray // will not work
13181: SIRegister_TALFBXPoolStream(CL);
13182: //PALFBXBlobData', '^PALFBXBlobData // will not work

```

```

13183: TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13184: //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13185: //TALFBXArrayDesc', 'TISCArryDesc
13186: //TALFBXBlobDesc', 'TISCblobDesc
13187: //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13188: //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13189: SIRegister_TALFBXSQLResult(CL);
13190: //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13191: SIRegister_TALFBXSQLParams(CL);
13192: //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13193: TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13194: +'atementType : TALFBXstatementType; end
13195: FindClass('TOBJECT'), TALFBXLibrary
13196: //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13197: TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary )
13198: //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13199: +'Excep : EALFBXExceptionClass)
13200: SIRegister_TALFBXLibrary(CL);
13201: 'calfBXDateOffset', 'LongInt'( 15018 );
13202: 'calfBXTIMECoef', 'LongInt'( 864000000 );
13203: //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13204: //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp );
13205: //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp ) : Double;
13206: Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word )
13207: Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13208: //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp );
13209: //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp );
13210: //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp );
13211: Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer ) : Integer
13212: Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord ) : Cardinal
13213: TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13214: TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13215: Function ALFBXSQLQuote( const name : AnsiString ) : AnsiString
13216: Function ALFBXSQLUnQuote( const name : AnsiString ) : AnsiString
13217: end;
13218:
13219: procedure SIRegister_ALFBXClient(CL: TPPascalCompiler);
13220: begin
13221:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13222:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13223:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13224: +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13225: +'teger; First : Integer; CacheThreshold : Integer; end
13226:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13227:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13228:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13229:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13230: +'_writes : int64; page_fetches : int64; page_marks : int64; end
13231:   SIRegister_TALFBXClient(CL);
13232:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13233:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13234:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13235:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13236:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13237:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13238:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13239:   SIRegister_TALFBXConnectionPoolClient(CL);
13240:   SIRegister_TALFBXEventThread(CL);
13241:   Function AlMySqlClientSlashedStr( const Str : AnsiString ) : AnsiString
13242: end;
13243:
13244: procedure SIRegister_ovcBidi(CL: TPPascalCompiler);
13245: begin
13246:   _OSVERSIONINFOA = record
13247:     dwOSVersionInfoSize: DWORD;
13248:     dwMajorVersion: DWORD;
13249:     dwMinorVersion: DWORD;
13250:     dwBuildNumber: DWORD;
13251:     dwPlatformId: DWORD;
13252:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13253:   end;
13254:   TOSversionInfoA', '_OSVERSIONINFOA
13255:   TOSversionInfo', 'TOSVersionInfoA
13256: 'WS_EX_RIGHT', 'LongWord')($00001000);
13257: 'WS_EX_LEFT', 'LongWord')($00000000);
13258: 'WS_EX_RTLREADING', 'LongWord')($00002000);
13259: 'WS_EX_LTRREADING', 'LongWord')($00000000);
13260: 'WS_EX_LEFTSCROLLBAR', 'LongWord')($00004000);
13261: 'WS_EX_RIGHTSCROLLBAR', 'LongWord')($00000000);
13262: Function SetProcessDefaultLayout( dwDefaultLayout : DWORD ) : BOOL
13263: 'LAYOUT_RTL', 'LongWord')($00000001);
13264: 'LAYOUT_BTT', 'LongWord')($00000002);
13265: 'LAYOUT_VBH', 'LongWord')($00000004);
13266: 'LAYOUT_BITMAPORIENTATIONPRESERVED', 'LongWord')($00000008);
13267: 'NOMIRRORBITMAP', 'LongWord')($00000000);
13268: Function SetLayout( dc : HDC; dwLayout : DWORD ) : DWORD
13269: Function GetLayout( dc : hdc ) : DWORD
13270: Function IsBidi : Boolean
13271: Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo ) : BOOL

```

```

13272: Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13273: Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13274: Function GetPriorityClass( hProcess : THandle) : DWORD
13275: Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13276: Function CloseClipboard : BOOL
13277: Function GetClipboardSequenceNumber : DWORD
13278: Function GetClipboardOwner : HWND
13279: Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13280: Function GetClipboardViewer : HWND
13281: Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13282: Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13283: Function GetClipboardData( uFormat : UINT) : THandle
13284: Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13285: Function CountClipboardFormats : Integer
13286: Function EnumClipboardFormats( format : UINT) : UINT
13287: Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13288: Function EmptyClipboard : BOOL
13289: Function IsClipboardFormatAvailable( format : UINT) : BOOL
13290: Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13291: Function GetOpenClipboardWindow : HWND
13292: Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13293: Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13294: Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13295: Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13296: Function SetDlgItemText( hDlg : HWND; nIDButton : Integer; lpString : PChar) : BOOL
13297: Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13298: Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13299: Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13300: Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13301: end;
13302:
13303: procedure SIRегистre_DXPUtils(CL: TPSPascalCompiler);
13304: begin
13305:   Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13306:   Function GetTemporaryFilesPath : String
13307:   Function GetTemporaryFileName : String
13308:   Function FindFileInPaths( const fileName, paths : String) : String
13309:   Function PathsToString( const paths : TStrings) : String
13310:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13311: //Function MacroExpandPath( const aPath : String) : String
13312: end;
13313:
13314: procedure SIRегистre_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13315: begin
13316:   SIRегистre_TALMultiPartBaseContent(CL);
13317:   SIRегистre_TALMultiPartBaseContents(CL);
13318:   SIRегистre_TALMultiPartBaseStream(CL);
13319:   SIRегистre_TALMultiPartBaseEncoder(CL);
13320:   SIRегистre_TALMultiPartBaseDecoder(CL);
13321:   Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13322:   Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13323:   Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13324: end;
13325:
13326: procedure SIRегистre_SmallUtils(CL: TPSPascalCompiler);
13327: begin
13328:   TdriveSize', 'record FreeS : Int64; TotalsS : Int64; end
13329:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In
13330:   +teger; WinMinorVersion : Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13331:   Function aAllocPadedMem( Size : Cardinal) : TObject
13332:   Procedure aFreePadedMem( var P : TObject);
13333:   Procedure aFreePadedMem( var P : PChar);
13334:   Function aCheckPadedMem( P : Pointer) : Byte
13335:   Function aGetPadMemSize( P : Pointer) : Cardinal
13336:   Function aAllocMem( Size : Cardinal) : Pointer
13337:   Function aStrLen( const Str : PChar) : Cardinal
13338:   Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13339:   Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13340:   Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13341:   Function aStrEnd( const Str : PChar) : PChar
13342:   Function aStrScan( const Str : PChar; aChr : Char) : PChar
13343:   Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13344:   Function aPCharLength( const Str : PChar) : Cardinal
13345:   Function aPCharUpper( Str : PChar) : PChar
13346:   Function aPCharLower( Str : PChar) : PChar
13347:   Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13348:   Function aLastDelimiter( const Delimiters, S : String) : Integer
13349:   Function aCopyTail( const S : String; Len : Integer) : String
13350:   Function aInt2Thos( I : Int64) : String
13351:   Function aUpperCase( const S : String) : String
13352:   Function aLowerCase( const S : string) : String
13353:   Function aCompareText( const S1, S2 : string) : Integer
13354:   Function aSameText( const S1, S2 : string) : Boolean
13355:   Function aInt2Str( Value : Int64) : String
13356:   Function aStr2Int( const Value : String) : Int64
13357:   Function aStr2IntDef( const S : string; Default : Int64) : Int64
13358:   Function aGetFileExt( const FileName : String) : String
13359:   Function aGetFilePath( const FileName : String) : String
13360:   Function a.GetFileName( const FileName : String) : String

```

```

13361: Function aChangeExt( const FileName, Extension : String ) : String
13362: Function aAdjustLineBreaks( const S : string ) : string
13363: Function aGetWindowStr( WinHandle : HWND ) : String
13364: Function aDiskSpace( Drive : String ) : TdriveSize
13365: Function aFileExists( FileName : String ) : Boolean
13366: Function aFileSize( FileName : String ) : Int64
13367: Function aDirectoryExists( const Name : string ) : Boolean
13368: Function aSysErrorMessage( ErrorCode : Integer ) : string
13369: Function aShortPathName( const LongName : string ) : string
13370: Function aGetWindowVer : TWinVerRec
13371: procedure InitDriveSpacePtr;
13372: end;
13373:
13374: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13375: begin
13376:   aZero', 'LongInt'( 0 );
13377:   'makeappDEF', 'LongInt'( - 1 );
13378:   'CS_VREDRAW', 'LongInt'( DWORD ( 1 ) );
13379:   'CS_HREDRAW', 'LongInt'( DWORD ( 2 ) );
13380:   'CS_KEYCVTWINDOW', 'LongInt'( 4 );
13381:   'CS_DBLCLKS', 'LongInt'( 8 );
13382:   'CS_OWNDC', 'LongWord')( $20 );
13383:   'CS_CLASSDC', 'LongWord')( $40 );
13384:   'CS_PARENTDC', 'LongWord')( $80 );
13385:   'CS_NOKEYCWT', 'LongWord')( $100 );
13386:   'CS_NOCLOSE', 'LongWord')( $200 );
13387:   'CS_SAVEBITS', 'LongWord')( $800 );
13388:   'CS_BYTEALIGNCLIENT', 'LongWord')( $1000 );
13389:   'CS_BYTEALIGNWINDOW', 'LongWord')( $2000 );
13390:   'CS_GLOBALCLASS', 'LongWord')( $4000 );
13391:   'CS_IME', 'LongWord')( $10000 );
13392:   'CS_DROPSHADOW', 'LongWord')( $20000 );
13393:   //TPanelFunc', 'TPanelFunc // will not work
13394:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13395:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13396:   TFontLooks', 'set of TFontLook
13397:   TMessagefunc', 'function(hWnd,iMsg,wParam:lParam:Integer):Integer
13398:   Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13399:   Function SetWinClassO( const ClassName : String; pMessFunc : Tobject; wcStyle : Integer): Word
13400:   Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer ) : Word
13401:   Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13402:   Procedure RunMsgLoop( Show : Boolean )
13403:   Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13404:   Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
ID_Number:Cardinal;hFont:Int):Int;
13405:   Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13406:   Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13407:   Function MakePanel(Left,Top,Width,Height,
hParent:Int;hndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13408:   Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13409:   Function id4menu( a, b : Byte; c : Byte; d : Byte ) : Cardinal
13410:   Procedure DoInitMakeApp //set first to init formclasscontrol!
13411: end;
13412:
13413: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13414: begin
13415:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13416:   +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13417:   TScreenSaverOptions', 'set of TScreenSaverOption
13418:   'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13419:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND )
13420:   SIRegister_TScreensaver(CL);
13421:   //Procedure Register
13422:   Procedure SetScreenSaverPassword
13423: end;
13424:
13425: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13426: begin
13427:   FindClass('TOBJECT'),'TXCollection
13428:   SIRegister_EFilerException(CL);
13429:   SIRegister_TXCollectionItem(CL);
13430:   //TXCollectionItemClass', 'class of TXCollectionItem
13431:   SIRegister_TXCollection(CL);
13432:   Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent )
13433:   Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent )
13434:   Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass )
13435:   Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass )
13436:   Function FindXCollectionItemClass( const className : String ) : TXCollectionItemClass
13437:   Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass ) : TList
13438: end;
13439:
13440: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13441: begin
13442:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond, mtcmArbitrary);
13443:   Procedure xglMapTexCoordToNull
13444:   Procedure xglMapTexCoordToMain
13445:   Procedure xglMapTexCoordToSecond
13446:   Procedure xglMapTexCoordToDual

```

```

13447: Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13448: Procedure xglMapTexCoordToArbitrary( const bitWiseUnits : Cardinal);
13449: Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13450: Procedure xglBeginUpdate
13451: Procedure xglEndUpdate
13452: Procedure xglPushState
13453: Procedure xglPopState
13454: Procedure xglForbidSecondTextureUnit
13455: Procedure xglAllowSecondTextureUnit
13456: Function xglGetBitWiseMapping : Cardinal
13457: end;
13458:
13459: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13460: begin
13461:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13462:   TBaseListOptions', 'set of TBaseListOption
13463:   SIRegister_TBaseList(CL);
13464:   SIRegister_TBaseVectorList(CL);
13465:   SIRegister_TAffineVectorList(CL);
13466:   SIRegister_TVectorList(CL);
13467:   SIRegister_TTexPointList(CL);
13468:   SIRegister_TXIntegerList(CL);
13469:   //PSingleArrayList', '^TSingleArrayList // will not work
13470:   SIRegister_TSingleList(CL);
13471:   SIRegister_TByteList(CL);
13472:   SIRegister_TQuaternionList(CL);
13473:   Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList );
13474:   Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList );
13475:   Procedure FastQuickSortLists(startIndex,
endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13476: end;
13477:
13478: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13479: begin
13480:   Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList );
13481:   Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList );
13482:   Procedure ConvertStripToList2(const strip:TAffineVectorList;const
indices:TIntegerList;list:TAffineVectorList);
13483:   Procedure ConvertIndexedListToList( const data:TAffineVectlist;const
indices:TIntegerList;list:TAffineVectorList );
13484:   Function BuildVectorCountOptimizedIndices( const vertices:TAffineVectorList; const
normals:TAffineVectorList; const texCoords : TAffineVectorList ) : TIntegerList
13485:   Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList );
13486:   Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList );
13487:   Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList )
13488:   Function RemapIndicesToIndicesMap( remapIndices : TIntegerList ) : TIntegerList
13489:   Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList )
13490:   Procedure RemapIndices( indices, indicesMap : TIntegerList )
13491:   Procedure UnifyTrianglesWinding( indices : TIntegerList )
13492:   Procedure InvertTrianglesWinding( indices : TIntegerList )
13493:   Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList ) : TAffineVectorList
13494:   Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
edgesTriangles : TIntegerList ) : TIntegerList
13495:   Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single )
13496:   Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean);
TPersistentObjectList;
13497:   Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer )
13498:   Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent )
13499: end;
13500:
13501: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13502: begin
13503:   Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte )
13504:   Procedure FreeMemAndNil( var P : TObject )
13505:   Function PCharOrNil( const S : string ) : PChar
13506:   SIRegister_TJclReferenceMemoryStream(CL);
13507:   FindClass('TOBJECT'), EJclVMTError
13508:   {Function GetVirtualMethodCount( AClass : TClass ) : Integer
13509:   Function GetVirtualMethod( AClass : TClass; const Index : Integer ) : Pointer
13510:   Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer )
13511:   PDynamicIndexList', 'TDynamicIndexList // will not work
13512:   PDynamicAddressList', '^TDynamicAddressList // will not work
13513:   Function GetDynamicMethodCount( AClass : TClass ) : Integer
13514:   Function GetDynamicIndexList( AClass : TClass ) : PDynamicIndexList
13515:   Function GetDynamicAddressList( AClass : TClass ) : PDynamicAddressList
13516:   Function HasDynamicMethod( AClass : TClass; Index : Integer ) : Boolean
13517:   Function GetDynamicMethod( AClass : TClass; Index : Integer ) : Pointer
13518:   Function GetInitTable( AClass : TClass ) : PTypeInfo
13519:   PFieldEntry', '^TFieldEntry // will not work
13520:   TFieldEntry', 'record Offset : Integer; IDX : Word; Name : ShortString; end
13521:   Function JIsClass( Address : Pointer ) : Boolean
13522:   Function JIsObject( Address : Pointer ) : Boolean
13523:   Function GetImplementorOfInterface( const I : IInterface ) : TObject
13524:   TdigitCount', 'Integer
13525:   SIRegister_TJclNumericFormat(CL);
13526:   Function JIntToStrZeroPad( Value, Count : Integer ) : AnsiString
13527:   TTextHandler', 'Procedure ( const Text : string )
13528:   // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
```

```

13529: Function JExecute( const
13530:   CommandLine:string;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool ):Card;
13531: Function JExecute1( const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool ):Cardinal;
13532: Function ReadKey : Char //to and from the DOS console !
13532: TModuleHandle', 'HINST
13533: //TModuleHandle', 'Pointer
13534: 'INVALID_MODULEHANDLE_VALUE', LongInt'( TModuleHandle ( 0 ) );
13535: Function LoadModule( var Module : TModuleHandle; FileName : string ) : Boolean
13536: Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal ) : Boolean
13537: Procedure UnloadModule( var Module : TModuleHandle )
13538: Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string ) : Pointer
13539: Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean ) : Pointer
13540: Function ReadModuleData( Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal ):Boolean;
13541: Function WriteModuleData( Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal ):Boolean;
13542: FindClass('TOBJECT'),'EJclConversionError
13543: Function JStrToBoolean( const S : string ) : Boolean
13544: Function JBooleanToStr( B : Boolean ) : string
13545: Function JIntToBool( I : Integer ) : Boolean
13546: Function JBoolToInt( B : Boolean ) : Integer
13547: 'ListSeparator','String '
13548: 'ListSeparator1','String :
13549: Procedure ListAddItems( var List : string; const Separator, Items : string )
13550: Procedure ListIncludeItems( var List : string; const Separator, Items : string )
13551: Procedure ListRemoveItems( var List : string; const Separator, Items : string )
13552: Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer )
13553: Function ListItemCount( const List, Separator : string ) : Integer
13554: Function ListGetItem( const List, Separator : string; const Index : Integer ) : string
13555: Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13556: Function ListItemIndex( const List, Separator, Item : string ) : Integer
13557: Function SystemToObjectInstance : LongWord
13558: Function IsCompiledWithPackages : Boolean
13559: Function JJclGUIDToString( const GUID : TGUID ) : string
13560: Function JJclStringToGUID( const S : string ) : TGUID
13561: SIRegister_TJclIntfCriticalSection(CL);
13562: SIRegister_TJclSimpleLog(CL);
13563: Procedure InitSimpleLog( const ALogFileFileName : string )
13564: end;
13565:
13566: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13567: begin
13568:   FindClass('TOBJECT'),'EJclBorRADException
13569:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )'
13570:   TJclBorRADToolEdition', '( deOPEN, dePRO, desVR )'
13571:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )'
13572:   TJclBorRADToolPath', 'string
13573: 'SupportedDelphiVersions', LongInt'( 5 or 6 or 7 or 8 or 9 or 10 or 11 );
13574: 'SupportedBCBVersions', LongInt'( 5 or 6 or 10 or 11 );
13575: 'SupportedBDSVersions', LongInt'( 1 or 2 or 3 or 4 or 5 );
13576: BorRADToolRepositoryPagesSection', 'String 'Repository Pages
13577: BorRADToolRepositoryDialogsPage', 'String 'Dialogs
13578: BorRADToolRepositoryFormsPage', 'String 'Forms
13579: BorRADToolRepositoryProjectsPage', 'String 'Projects
13580: BorRADToolRepositoryDataModulesPage', 'String 'Data Modules
13581: BorRADToolRepositoryObjectType', 'String 'Type
13582: BorRADToolRepositoryFormTemplate', 'String 'FormTemplate
13583: BorRADToolRepositoryProjectTemplate', 'String 'ProjectTemplate
13584: BorRADToolRepositoryObjectName', 'String 'Name
13585: BorRADToolRepositoryObjectPage', 'String 'Page
13586: BorRADToolRepositoryObjectIcon', 'String 'Icon
13587: BorRADToolRepositoryObjectDescr', 'String 'Description
13588: BorRADToolRepositoryObjectAuthor', 'String 'Author
13589: BorRADToolRepositoryObjectAncestor', 'String 'Ancestor
13590: BorRADToolRepositoryObjectDesigner', 'String 'Designer
13591: BorRADToolRepositoryDesignerDfm', 'String 'dfm
13592: BorRADToolRepositoryDesignerXfm', 'String 'xmf
13593: BorRADToolRepositoryObjectNewForm', 'String 'DefaultNewForm
13594: BorRADToolRepositoryObjectMainForm', 'String 'DefaultMainForm
13595: SourceExtensionDelphiPackage', 'String '.dpk
13596: SourceExtensionBCBPackage', 'String '.bpk
13597: SourceExtensionDelphiProject', 'String '.dpr
13598: SourceExtensionBCBProject', 'String '.bpr
13599: SourceExtensionBDSProject', 'String '.bdsproj
13600: SourceExtensionDProject', 'String '.dproj
13601: BinaryExtensionPackage', 'String '.bpl
13602: BinaryExtensionLibrary', 'String '.dll
13603: BinaryExtensionExecutable', 'String '.exe
13604: CompilerExtensionDCP', 'String '.dcp
13605: CompilerExtensionBPI', 'String '.bpi
13606: CompilerExtensionLIB', 'String '.lib
13607: CompilerExtensionTDS', 'String '.tds
13608: CompilerExtensionMAP', 'String '.map
13609: CompilerExtensionDRC', 'String '.drc
13610: CompilerExtensionDEF', 'String '.def
13611: SourceExtensionCPP', 'String '.cpp
13612: SourceExtensionH', 'String '.h
13613: SourceExtensionPAS', 'String '.pas
13614: SourceExtensionDFM', 'String '.dfm
13615: SourceExtensionXFM', 'String '.xfm
13616: SourceDescriptionPAS', 'String 'Pascal source file

```

```

13617: SourceDescriptionCPP', 'String 'C++ source file
13618: DesignerVCL', 'String 'VCL
13619: DesignerCLX', 'String 'CLX
13620: ProjectTypePackage', 'String 'package
13621: ProjectTypeLibrary', 'String 'library
13622: ProjectTypeProgram', 'String 'program
13623: Personality32Bit', 'String '32 bit
13624: Personality64Bit', 'String '64 bit
13625: PersonalityDelphi', 'String 'Delphi
13626: PersonalityDelphiDotNet', 'String 'Delphi.net
13627: PersonalityBCB', 'String 'C++Builder
13628: PersonalityCSB', 'String 'C#Builder
13629: PersonalityVB', 'String 'Visual Basic
13630: PersonalityDesign', 'String 'Design
13631: PersonalityUnknown', 'String 'Unknown personality
13632: PersonalityBDS', 'String 'Borland Developer Studio
13633: DOFDirectoriesSection', 'String 'Directories
13634: DOFUnitOutputDirKey', 'String 'UnitOutputDir
13635: DOFSearchPathName', 'String 'SearchPath
13636: DOFConditionals', 'String 'Conditionals
13637: DOFLinkerSection', 'String 'Linker
13638: DOFPackagesKey', 'String 'Packages
13639: DOFCompilerSection', 'String 'Compiler
13640: DOFPackageNoLinkKey', 'String 'PackageNoLink
13641: DOFAdditionalSection', 'String 'Additional
13642: DOFOptionsKey', 'String 'Options
13643: TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13644: '+ pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13645: '+ bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13646: TJclBorPersonalities', 'set of TJclBorPersonality
13647: TJclBorDesigner', '( bdVCL, bdCLX )
13648: TJclBorDesigners', 'set of TJclBorDesigner
13649: TJclBorPlatform', '( bp32bit, bp64bit )
13650: FindClass('TOBJECT'), 'TJclBorRADToolInstallation
13651: SIRegister_TJclBorRADToolInstallationObject(CL);
13652: SIRegister_TJclBorLandOpenHelp(CL);
13653: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13654: TJclHelp2Objects', 'set of TJclHelp2Object
13655: SIRegister_TJclHelp2Manager(CL);
13656: SIRegister_TJclBorRADToolIDETool(CL);
13657: SIRegister_TJclBorRADToolIDEPackages(CL);
13658: SIRegister_IJclCommandLineTool(CL);
13659: FindClass('TOBJECT'), 'EJclCommandLineToolError
13660: SIRegister_TJclCommandLineTool(CL);
13661: SIRegister_TJclBorLandCommandLineTool(CL);
13662: SIRegister_TJclBCC32(CL);
13663: SIRegister_TJclDCC32(CL);
13664: TJclDCC', 'TJclDCC32
13665: SIRegister_TJclBpr2Mak(CL);
13666: SIRegister_TJclBorLandMake(CL);
13667: SIRegister_TJclBorRADToolPalette(CL);
13668: SIRegister_TJclBorRADToolRepository(CL);
13669: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
13670: TCommandLineTools', 'set of TCommandLineTool
13671: //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13672: SIRegister_TJclBorRADToolInstallation(CL);
13673: SIRegister_TJclBCBInstallation(CL);
13674: SIRegister_TJclDelphiInstallation(CL);
13675: SIRegister_TJclDCCIL(CL);
13676: SIRegister_TJclBDSInstallation(CL);
13677: TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation ) : Boolean
13678: SIRegister_TJclBorRADToolInstallations(CL);
13679: Function BPLFileName( const BPLPath, PackageFileName : string ) : string
13680: Function BinaryFileName( const OutputPath, ProjectFileName : string ) : string
13681: Function IsDelphiPackage( const FileName : string ) : Boolean
13682: Function IsDelphiProject( const FileName : string ) : Boolean
13683: Function IsBCBPackage( const FileName : string ) : Boolean
13684: Function IsBCBProject( const FileName : string ) : Boolean
13685: Procedure GetDPRFileInfo( const DPRFileName:string;out BinaryExtensio:string;const LibSuffix:PString );
13686: Procedure GetBPRFileInfo( const BPRFileName:string;out BinaryFileName:string;const Descript:PString );
13687: Procedure GetDPKFileInfo( const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
Descript:PString;
13688: Procedure GetBPKFileInfo( const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
Descript:PString
13689: function SamePath(const Path1, Path2: string): Boolean;
13690: end;
13691:
13692: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13693: begin
13694:   'ERROR_NO_MORE_FILES', LongInt( 18 );
13695:   //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13696:   //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64 ) : Integer
13697:   //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13698:   'LPathSeparator','String '/'
13699:   'LDirDelimiter','String '
13700:   'LDirSeparator','String '
13701:   'JXPathDevicePrefix','String '\.\.
13702:   'JXPathSeparator','String \
13703:   'JXDirDelimiter','String \

```

```

13704: 'JXDirSeparator','String ';
13705: 'JXPathUncPrefix','String '\\
13706: 'faNormalFile','LongWord')($$00000080);
13707: //faUnixSpecific,' faSymLink';
13708: JXTCompactPath', '( cpCenter, cpEnd )
13709: _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13710: +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime : '
13711: +' TFileTime; nfileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13712: TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13713: WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13714:
13715: Function jxPathAddSeparator( const Path : string ) : string
13716: Function jxPathAddExtension( const Path, Extension : string ) : string
13717: Function jxPathAppend( const Path, Append : string ) : string
13718: Function jxPathBuildRoot( const Drive : Byte ) : string
13719: Function jxPathCanonicalize( const Path : string ) : string
13720: Function jxPathCommonPrefix( const Path1, Path2 : string ) : Integer
13721: Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13722: Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string )
13723: Function jxPathExtractFileDirName( const S : string ) : string
13724: Function jxPathExtractFileNameNoExt( const Path : string ) : string
13725: Function jxPathExtractPathDepth( const Path : string; Depth : Integer ) : string
13726: Function jxPathGetDepth( const Path : string ) : Integer
13727: Function jxPathGetLongName( const Path : string ) : string
13728: Function jxPathGetShortName( const Path : string ) : string
13729: Function jxPathGetLongName( const Path : string ) : string
13730: Function jxPathGetShortName( const Path : string ) : string
13731: Function jxPathGetRelativePath( Origin, Destination : string ) : string
13732: Function jxPathGetTempPath : string
13733: Function jxPathIsAbsolute( const Path : string ) : Boolean
13734: Function jxPathIsChild( const Path, Base : string ) : Boolean
13735: Function jxPathIsDiskDevice( const Path : string ) : Boolean
13736: Function jxPathIsUNC( const Path : string ) : Boolean
13737: Function jxPathRemoveSeparator( const Path : string ) : string
13738: Function jxPathRemoveExtension( const Path : string ) : string
13739: Function jxPathGetPhysicalPath( const LocalizedPath : string ) : string
13740: Function jxPathGetLocalizedPath( const PhysicalPath : string ) : string
13741: JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders )
13742: JxTFileListOptions', 'set of TFileListOption
13743: JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13744: TFileHandler', 'Procedure ( const FileName : string )
13745: TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec )
13746: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings ) : Boolean
13747: //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
13748: AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
13749: FileMatchFunc:TFileMatchFunc):Bool;
13750: Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13751: Function jxFileAttributesStr( const FileInfo : TSearchRec ) : string
13752: Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13753: Procedure jxEnumFiles(const Path:string;HandleFile:TFileHandlerEx;
13754: RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13755: Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
13756: IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13757: Procedure jxCreatEmptyFile( const FileName : string )
13758: Function jxCloseVolume( var Volume : THandle ) : Boolean
13759: Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean ) : Boolean
13760: Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13761: Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13762: Function jxDelTree( const Path : string ) : Boolean
13763: //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13764: Function jxDiskInDrive( Drive : Char ) : Boolean
13765: Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean ) : Boolean
13766: Function jxFolderCreateTemp( var Prefix : string ) : THandle
13767: Function jxFolderBackup( const FileName : string; Move : Boolean ) : Boolean
13768: Function jxFolderCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13769: Function jxFolderDelete( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
13770: Function jxFolderExists( const FileName : string ) : Boolean
13771: Function jxFolderMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13772: Function jxFolderRestore( const FileName : string ) : Boolean
13773: Function jxFolderGetBackupFileName( const FileName : string ) : string
13774: Function jxFolderIsBackupFileName( const FileName : string ) : Boolean
13775: Function jxFolderGetDisplayName( const FileName : string ) : string
13776: Function jxFolderGetSize( const FileName : string ) : Int64
13777: Function jxFolderGetTempName( const Prefix : string ) : string
13778: Function jxFolderGetType( const FileName : string ) : string
13779: Function jxFolderFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string ) : string
13780: Function jxFolderForceDirectories( Name : string ) : Boolean
13781: Function jxFolderGetDirectorySize( const Path : string ) : Int64
13782: Function jxFolderGetDriveTypeStr( const Drive : Char ) : string
13783: Function jxFolderGetFileAgeCoherence( const FileName : string ) : Boolean
13784: Procedure jxFolderGetFileAttributeList( const Items : TStrings; const Attr : Integer )
13785: Procedure jxFolderGetFileAttributeListEx( const Items : TStrings; const Attr : Integer )
13786: Function jxFolderGetFileInfo( const FileName : string; out FileInfo : TSearchRec ) : Boolean;
13787: Function jxFolderGetFileInformation( const FileName : string ) : TSearchRec;
13788: //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
13789: ResolveSymLinks:Boolean):Integer

```

```

13788: Function jxGetFileLastWrite( const FName : string ) : TFileTime;
13789: Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13790: Function jxGetFileLastAccess( const FName : string ) : TFileTime;
13791: Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13792: Function jxGetFileCreation( const FName : string ) : TFileTime;
13793: Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13794: Function jxGetFileLastWrite( const FName : string; out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13795: Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13796: Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean ) : Integer;
13797: Function jxGetFileLastAccess( const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13798: Function jxGetFileLastAccess1( const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13799: Function jxGetFileLastAccess2( const FName:string; ResolveSymLinks:Boolean): Integer;
13800: Function jxGetFileLastAttrChange( const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13801: Function jxGetFileLastAttrChange1( const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13802: Function jxGetModulePath( const Module : HMODULE ) : string;
13803: Function jxGetSizeOfFile( const FileName : string ) : Int64;
13805: Function jxGetSizeOfFile1( const FileInfo : TSearchRec ) : Int64;
13806: Function jxGetSizeOfFile2( Handle : THandle ) : Int64;
13807: Function jxGetStandardFileInfo( const FileName : string ) : TWin32FileAttributeData;
13808: Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean ) : Boolean;
13809: Function jxIsRootDirectory( const CanonicFileName : string ) : Boolean;
13810: Function jxLockVolume( const Volume : string; var Handle : THandle ) : Boolean;
13811: Function jxOpenVolume( const Drive : Char ) : THandle;
13812: Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime ) : Boolean;
13813: Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime ) : Boolean;
13814: Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime ) : Boolean;
13815: Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime ) : Boolean;
13816: Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime ) : Boolean;
13817: Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime ) : Boolean;
13818: Procedure jxShredfile( const FileName : string; Times : Integer );
13819: Function jxUnlockVolume( var Handle : THandle ) : Boolean;
13820: Function jxCreateSymbolicLink( const Name, Target : string ) : Boolean;
13821: Function jxSymbolicLinkTarget( const Name : string ) : string;
13822: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )';
13823: SIRegister_TJclCustomFileAttrMask(CL);
13824: SIRegister_TJclFileAttributeMask(CL);
13825: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS' +
13826: '+ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)';
13827: TFileSearchOptions', 'set of TFileSearchOption;
13828: TFileSearchTaskID', 'Integer;
13829: TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc' +
13830: '+hTaskID; const Aborted : Boolean)';
13831: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )';
13832: SIRegister_IJclFileEnumerator(CL);
13833: SIRegister_TJclFileVersionInfo(CL);
13834: Function JxFileVersionSearch : IJclFileEnumerator;
13835: JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched, ffPreRelease, ffPrivateBuild, ffSpecialBuild )';
13836: JxTFileFlags', 'set of TFileFlag;
13837: FindClass('TOBJECT'), 'EJclFileVersionInfoError';
13838: SIRegister_TJclFileVersionInfo(CL);
13839: Function jxOSIdToString( const OSId : DWORD ) : string;
13840: Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD ) : string;
13841: Function jxVersionResourceAvailable( const FileName : string ) : Boolean;
13842: TFileVersionFormat', '( vfMajorMinor, vfFull )';
13843: Function jxFormatVersionString( const HiV, LoV : Word ) : string;
13844: Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word ) : string;
13845: //Function FormatVersionString2( const FileInfo : TVSFixedFileInfo; VersionFormat:TFFileVersionFormat):str;
13846: //Procedure VersionExtractFileInfo(const FileInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13847: //Procedure VersionExtractProductInfo(const FileInfo:TVSFixedFileInfo;var Major,Minor,Build,
13848: Revision:Word);
13849: //Function VersionFixedFileInfo( const FileName : string; var FileInfo : TVSFixedFileInfo ) : Boolean;
13849: Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFFileVersionFormat; const
13850: NotAvailableText : string ) : string;
13850: SIRegister_TJclTempFileStream(CL);
13851: FindClass('TOBJECT'), 'TJclCustomFileMapping';
13852: SIRegister_TJclFileMapView(CL);
13853: TJclFileMappingRoundOffset', '( rvDown, rvUp )';
13854: SIRegister_TJclCustomFileMapping(CL);
13855: SIRegister_TJclFileMapping(CL);
13856: SIRegister_TJclSwapFileMapping(CL);
13857: SIRegister_TJclFileMappingStream(CL);
13858: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )';
13859: //PPCharArray', '^TPCharArray // will not work
13860: SIRegister_TJclMappedTextReader(CL);
13861: SIRegister_TJclFileMaskComparator(CL);
13862: FindClass('TOBJECT'), 'EJclPathError';
13863: FindClass('TOBJECT'), 'EJclFileUtilsError';
13864: FindClass('TOBJECT'), 'EJclTempFileStreamError';
13865: FindClass('TOBJECT'), 'EJclTempFileStreamError';
13866: FindClass('TOBJECT'), 'EJclFileMappingError';
13867: FindClass('TOBJECT'), 'EJclFileMappingViewError';
13868: Function jxPathGetLongName2( const Path : string ) : string;
13869: Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean;
13870: Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstfileName : string ) : Boolean;
13871: Function jxWin32BackupFile( const FileName : string; Move : Boolean ) : Boolean;
13872: Function jxWin32RestoreFile( const FileName : string ) : Boolean;
13873: Function jxSamePath( const Path1, Path2 : string ) : Boolean;
13874: Procedure jxPathListAddItems( var List : string; const Items : string );

```

```

13875: Procedure jxPathListIncludeItems( var List : string; const Items : string)
13876: Procedure jxPathListDeleteItems( var List : string; const Items : string)
13877: Procedure jxPathListDeleteItem( var List : string; const Index : Integer)
13878: Function jxPathListItemCount( const List : string) : Integer
13879: Function jxPathListGetItem( const List : string; const Index : Integer) : string
13880: Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13881: Function jxPathListItemIndex( const List, Item : string) : Integer
13882: Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13883: Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13884: Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
  AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13885: Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
  AllowedPrefixCharacters : string) : Integer
13886: end;
13887:
13888: procedure SIRRegister_FileUtil(CL: TPSCompiler);
13889: begin
13890:   'UTF8FileHeader','String #$ef#$bb#$bf';
13891:   Function lCompareFilenames( const Filenam1, Filenam2 : string) : integer
13892:   Function lCompareFilenamesIgnoreCase( const Filenam1, Filenam2 : string) : integer
13893:   Function lCompareFilenames( const Filenam1, Filenam2 : string; ResolveLinks : boolean) : integer
13894:   Function lCompareFilenames(Filenam1:PChar;Len1:int;Filenam2:PChar;Len2:int;ResolveLinks:boolean):int;
13895:   Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13896:   Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13897:   Function lFilenameIsUnixAbsolute( const Thefilename : string) : boolean
13898:   Procedure lCheckIfFileIsExecutable( const Afilename : string)
13899:   Procedure lCheckIfFileIsSymlink( const Afilename : string)
13900:   Function lFileIsReadable( const Afilename : string) : boolean
13901:   Function lFileIsWritable( const Afilename : string) : boolean
13902:   Function lFileIsText( const Afilename : string) : boolean
13903:   Function lFileIsText( const Afilename : string; out FileReadable : boolean) : boolean
13904:   Function lFileIsExecutable( const Afilename : string) : boolean
13905:   Function lFileIsSymlink( const Afilename : string) : boolean
13906:   Function lFileIsHardLink( const Afilename : string) : boolean
13907:   Function lFileSize( const Filenam : string) : int64;
13908:   Function lGetFileDescription( const Afilename : string) : string
13909:   Function lReadAllLinks( const Filenam : string; ExceptionOnError : boolean) : string
13910:   Function lTryReadAllLinks( const Filenam : string) : string
13911:   Function lDirPathExists( const FileName : String) : Boolean
13912:   Function lForceDirectory( DirectoryName : string) : boolean
13913:   Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13914:   Function lProgramDirectory : string
13915:   Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13916:   Function lExtractFileNameOnly( const Afilename : string) : string
13917:   Function lExtractFileNameWithoutExt( const Afilename : string) : string
13918:   Function lCompareFileExt( const Filenam, Ext : string; CaseSensitive : boolean) : integer;
13919:   Function lCompareFileExt( const Filenam, Ext : string) : integer;
13920:   Function lFilenameIsPascalUnit( const Filenam : string) : boolean
13921:   Function lAppendPathDelim( const Path : string) : string
13922:   Function lChompPathDelim( const Path : string) : string
13923:   Function lTrimFilename( const Afilename : string) : string
13924:   Function lCleanAndExpandFilename( const Filenam : string) : string
13925:   Function lCleanAndExpandDirectory( const Filenam : string) : string
13926:   Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13927:   Function lCreateRelativePath( const Filenam, BaseDirectory : string; UsePointDirectory : boolean;
  AlwaysRequireSharedBaseFolder : Boolean) : string
13928:   Function lCreateAbsolutePath( const Filenam, BaseDirectory : string) : string
13929:   Function lFileIsInPath( const Filenam, Path : string) : boolean
13930:   Function lFileIsInDirectory( const Filenam, Directory : string) : boolean
13931:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13932:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13933:   'AllDirectoryEntriesMask','String '*
13934:   Function l GetAllFilesMask : string
13935:   Function lGetExeExt : string
13936:   Function lSearchFileInPath( const Filenam, basePath, SearchPath, Delimiter : string; Flags :
  TSearchFileInPathFlags) : string
13937:   Function lSearchAllFilesInPath( const Filenam, basePath, SearchPath, Delimiter:string;Flags :
  TSearchFileInPathFlags) : TStrings
13938:   Function lFindDiskFilename( const Filenam : string) : string
13939:   Function lFindDiskFileCaseInsensitive( const Filenam : string) : string
13940:   Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13941:   Function lGetDarwinSystemFilename( Filenam : string) : string
13942:   SIRRegister_TFileIterator(CL);
13943:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13944:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13945:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13946:   SIRRegister_TFileSearcher(CL);
13947:   Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13948:   Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13949: // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13950: // TCopyFileFlags', 'set of TCopyFileFlag
13951:   Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13952:   Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13953:   Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13954:   Function lReadFileToString( const Filenam : string) : string
13955:   Function lGetTempFilename( const Directory, Prefix : string) : string
13956:   {Function NeedRTLAnsi : boolean
13957:   Procedure SetNeedRTLAnsi( NewValue : boolean)
13958:   Function UTF8ToSys( const s : string) : string

```

```

13959: Function SysToUTF8( const s : string ) : string
13960: Function ConsoleToUTF8( const s : string ) : string
13961: Function UTF8ToConsole( const s : string ) : string
13962: Function FileExistsUTF8( const FileName : string ) : boolean
13963: Function FileAgeUTF8( const FileName : string ) : Longint
13964: Function DirectoryExistsUTF8( const Directory : string ) : Boolean
13965: Function ExpandFileNameUTF8( const FileName : string ) : string
13966: Function ExpandUNCfileNameUTF8( const FileName : string ) : string
13967: Function ExtractShortPathNameUTF8( const FileName : String ) : String
13968: Function FindFirstUTF8( const Path: string; Attr : Longint; out Rslt : TSearchRec ) : Longint
13969: Function FindNextUTF8( var Rslt : TSearchRec ) : Longint
13970: Procedure FindCloseUTF8( var F : TSearchrec )
13971: Function FileSetDateUTF8( const FileName : String; Age : Longint ) : Longint
13972: Function FileGetAttrUTF8( const FileName : String ) : Longint
13973: Function FileSetAttrUTF8( const FileName : String; Attr : longint ) : Longint
13974: Function DeleteFileUTF8( const FileName : String ) : Boolean
13975: Function RenameFileUTF8( const OldName, NewName : String ) : Boolean
13976: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean ) : String
13977: Function FileIsReadOnlyUTF8( const FileName : String ) : Boolean
13978: Function GetCurrentDirUTF8 : String
13979: Function SetCurrentDirUTF8( const NewDir : String ) : Boolean
13980: Function CreateDirUTF8( const NewDir : String ) : Boolean
13981: Function RemoveDirUTF8( const Dir : String ) : Boolean
13982: Function ForceDirectoriesUTF8( const Dir : string ) : Boolean
13983: Function FileOpenUTF8( const FileName : string; Mode : Integer ) : THandle
13984: Function FileCreateUTF8( const FileName : string ) : THandle;
13985: Function FileCreateUTF8l( const FileName : string; Rights : Cardinal ) : THandle;
13986: Function ParamStrUTF8( Param : Integer ) : string
13987: Function GetEnvironmentStringUTF8( Index : Integer ) : string
13988: Function GetEnvironmentVariableUTF8( const EnvVar : string ) : String
13989: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean ) : string
13990: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean ) : string
13991: Function SysErrorMessageUTF8( ErrorCode : Integer ) : String
13992: end;
13993:
13994: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13995: begin
13996:   //VK_F23 = 134;
13997:   //{$EXTERNALSYM VK_F24}
13998:   //VK_F24 = 135;
13999:   TVirtualKeyCode', 'Integer
14000:   'VK_MOUSEWHEELUP', 'integer'(134);
14001:   'VK_MOUSEWHEELDOWN', 'integer'(135);
14002:   Function glIsKeyDown( c : Char ) : Boolean;
14003:   Function glIsKeyDown1( vk : TVirtualKeyCode ) : Boolean;
14004:   Function glKeyPressed( minVkeyCode : TVirtualKeyCode ) : TVirtualKeyCode
14005:   Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode ) : String
14006:   Function glKeyNameToVirtualKeyCode( const keyName : String ) : TVirtualKeyCode
14007:   Function glCharToVirtualKeyCode( c : Char ) : TVirtualKeyCode
14008:   Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer )
14009: end;
14010:
14011: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14012: begin
14013:   TGLPoint', 'TPoint
14014:   //PGLPoint', '^TGLPoint // will not work
14015:   TGLRect', 'TRect
14016:   //PGLRect', '^TGLRect // will not work
14017:   TDelphiColor', 'TColor
14018:   TGLPicture', 'TPicture
14019:   TGLGraphic', 'TGraphic
14020:   TGLBitmap', 'TBitmap
14021:   //TGraphicClass', 'class of TGraphic
14022:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14023:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
14024:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
14025:     + 'Button; Shift : TShiftState; X, Y : Integer )
14026:   TGLMouseMoveEvent', 'TMouseEvent
14027:   TGLKeyEvent', 'TKeyEvent
14028:   TGLKeyPressEvent', 'TKeyPressEvent
14029:   EGLOSError', 'EWin32Error
14030:   EGLOSError', 'EWin32Error
14031:   EGLOSError', 'EOSError
14032:   'gl$AllFilter', 'string'All // $AllFilter
14033:   Function GLPoint( const x, y : Integer ) : TGLPoint
14034:   Function GLRGB( const r, g, b : Byte ) : TColor
14035:   Function GLColorToRGB( color : TColor ) : TColor
14036:   Function GLGetRValue( rgb : DWORD ) : Byte
14037:   Function GLGetGValue( rgb : DWORD ) : Byte
14038:   Function GLGetBValue( rgb : DWORD ) : Byte
14039:   Procedure GLInitWinColors
14040:   Function GLRect( const aLeft, aTop, aRight, aBottom : Integer ) : TGLRect
14041:   Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer )
14042:   Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect )
14043:   Procedure GLInformationDlg( const msg : String )
14044:   Function GLQuestionDlg( const msg : String ) : Boolean
14045:   Function GLInputDlg( const msg : String ) : String
14046:   Function GLSavePictureDialog( var aFileName : String; const aTitle : String ) : Boolean
14047:   Function GLOpenPictureDialog( var aFileName : String; const aTitle : String ) : Boolean

```

```

14048: Function GLApplicationTerminated : Boolean
14049: Procedure GLRaiseLastOSError
14050: Procedure GLFreeAndNil( var anObject: TObject)
14051: Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
14052: Function GLGetCurrentColorDepth : Integer
14053: Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14054: Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14055: Procedure GLSleep( length : Cardinal)
14056: Procedure GLQueryPerformanceCounter( var val : Int64)
14057: Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14058: Function GLStartPrecisionTimer : Int64
14059: Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14060: Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14061: Function GLRTSC : Int64
14062: Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14063: Function GLOKMessageBox( const Text, Caption : string) : Integer
14064: Procedure GLShowHTMLUrl( Url : String)
14065: Procedure GLShowCursor( AShow : boolean)
14066: Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14067: Procedure GLGetCursorPos( var point : TGLPoint)
14068: Function GLGetScreenWidth : integer
14069: Function GLGetScreenHeight : integer
14070: Function GLGetTickCount : int64
14071: function RemoveSpaces(const str : String) : String;
14072: TNoramlMapSpace', ' nmsObject, nmsTangent )
14073: Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList:Colors:TVectorList)
14074: Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14075: Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals : TAffineVectorList; Colors : TVectorList)
14076: Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
HiTexCoords:TAffineVectorList):TGLBitmap
14077: Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14078: end;
14079:
14080: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14081: begin
14082:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14083: // PGLStarRecord', '^TGLStarRecord // will not work
14084: Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14085: Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14086: Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14087: end;
14088:
14089:
14090: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14091: begin
14092:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14093: //PAABB', '^TAABB // will not work
14094: TBSphere', 'record Center : TAffineVector; Radius : single; end
14095: TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14096: TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14097: Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox ) : THmgBoundingBox
14098: Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14099: Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14100: Procedure SetAABB( var bb : TAABB; const v : TVector)
14101: Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14102: Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14103: Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14104: Function BBMinX( const c : THmgBoundingBox ) : Single
14105: Function BBMaxX( const c : THmgBoundingBox ) : Single
14106: Function BBMinY( const c : THmgBoundingBox ) : Single
14107: Function BBMaxY( const c : THmgBoundingBox ) : Single
14108: Function BBMinZ( const c : THmgBoundingBox ) : Single
14109: Function BBMaxZ( const c : THmgBoundingBox ) : Single
14110: Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14111: Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14112: Function AABBIntersection( const aabb1, aabb2 : TAABB ) : TAABB
14113: Function BBTAAABB( const aBB : THmgBoundingBox ) : TAABB
14114: Function AABBTaaBB( const anAABB : TAABB ) : THmgBoundingBox;
14115: Function AABBTaaB1( const anAABB : TAABB; const m : TMatrix ) : THmgBoundingBox;
14116: Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14117: Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14118: Function IntersectAABBS( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix ) : Boolean;
14119: Function IntersectAABBSAbsoluteXY( const aabb1, aabb2 : TAABB ) : Boolean
14120: Function IntersectAABBSAbsoluteXZ( const aabb1, aabb2 : TAABB ) : Boolean
14121: Function IntersectAABBSAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14122: Function AABBFitInAABBAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14123: Function PointInAABB( const p : TAffineVector; const aabb : TAABB ) : Boolean;
14124: Function PointInAABB1( const p : TVector; const aabb : TAABB ) : Boolean;
14125: Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB ) : boolean
14126: Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector ) : boolean
14127: Procedure ExtractAABB_corners( const AABB : TAABB; var AABBCorners : AABBCorners)
14128: Procedure AABBTaaBSphere( const AABB : TAABB; var BSphere : TBSphere)
14129: Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14130: Function BSphereToAABB1( const center : TAffineVector; radius : Single ) : TAABB;
14131: Function BSphereToAABB2( const center : TVector; radius : Single ) : TAABB;
14132: Function AABBContainsAABB( const mainAABB, testAABB : TAABB ) : TSpaceContains
14133: Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB ) : TSpaceContains

```

```

14134: Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere ) : TSpaceContains
14135: Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere ) : TSpaceContains
14136: Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
testBSphere:TBSphere):TSpaceContains
14137: Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere ) : TSpaceContains
14138: Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB ) : TSpaceContains
14139: Function ClipToAABB( const v : TAffineVector; const AABB : TAABB ) : TAffineVector
14140: Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere ) : boolean
14141: Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single )
14142: Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
viewportSizeY:Int):TClipRect
14143: end;
14144:
14145: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14146: begin
14147: Procedure Cylindrical_Cartesian( const r, theta, zl : single; var x, y, z : single );
14148: Procedure Cylindrical_Cartesian1( const r, theta, zl : double; var x, y, z : double );
14149: Procedure Cylindrical_Cartesian2( const r,theta,zl : single; var x, y, z : single; var ierr : integer );
14150: Procedure Cylindrical_Cartesian3( const r,theta,zl : double; var x, y, z : double; var ierr : integer );
14151: Procedure Cartesian_Cylindrical( const x, y, zl : single; var r, theta, z : single );
14152: Procedure Cartesian_Cylindrical1( const x, y, zl : double; var r, theta, z : double );
14153: Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single );
14154: Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double );
14155: Procedure Spherical_Cartesian2( const r,theta,phi : single; var x, y, z : single; var ierr : integer );
14156: Procedure Spherical_Cartesian3( const r,theta,phi : double; var x, y, z : double; var ierr : integer );
14157: Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single );
14158: Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single );
14159: Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double );
14160: Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single );
14161: Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double );
14162: Procedure ProlateSpheroidal_Cartesian2( const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer );
14163: Procedure ProlateSpheroidal_Cartesian3( const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer );
14164: Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single );
14165: Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double );
14166: Procedure OblateSpheroidal_Cartesian2( const xi,eta,phi,a:single;var x,y,z:single;var ierr:integer );
14167: Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer );
14168: Procedure BipolarCylindrical_Cartesian( const u, v, zl, a : single; var x, y, z : single );
14169: Procedure BipolarCylindrical_Cartesian1( const u, v, zl, a : double; var x, y, z : double );
14170: Procedure BipolarCylindrical_Cartesian2( const u,v,zl,a : single;var x,y,z:single; var ierr : integer );
14171: Procedure BipolarCylindrical_Cartesian3( const u,v,zl,a : double;var x,y,z:double; var ierr : integer );
14172: end;
14173:
14174: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14175: begin
14176: 'EPSILON','Single').setExtended( 1e-40 );
14177: 'EPSILON2','Single').setExtended( 1e-30 ); }
14178: TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14179: + 'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14180: THmgPlane', 'TVector
14181: TDoubleHmgPlane', 'THomogeneousDblVector
14182: {TTtransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14183: +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14184: +' , ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14185: TSingleArray', 'array of Single
14186: TTtransformations', 'array [0..15] of Single)
14187: TPackedRotationMatrix', 'array [0..2] of Smallint)
14188: TVertex', 'TAffineVector
14189: //TVectorGL', 'THomogeneousFltVector
14190: //TMatrixGL', 'THomogeneousFltMatrix
14191: // TPackedRotationMatrix = array [0..2] of SmallInt;
14192: Function glTexPointMake( const s, t : Single ) : TTexPoint
14193: Function glAffineVectorMake( const x, y, z : Single ) : TAffineVector;
14194: Function glAffineVectorMake1( const v : TVectorGL ) : TAffineVector;
14195: Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single );
14196: Procedure glSetVector( var v : TAffineVector; const x, y, z : Single );
14197: Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL );
14198: Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector );
14199: Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector );
14200: Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL );
14201: Function glVectorMake( const v : TAffineVector; w : Single ) : TVectorGL;
14202: Function glVectorMake1( const x, y, z : Single; w : Single ) : TVectorGL;
14203: Function glPointMake( const x, y, z : Single ) : TVectorGL;
14204: Function glPointMake1( const v : TAffineVector ) : TVectorGL;
14205: Function glPointMake2( const v : TVectorGL ) : TVectorGL;
14206: Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single );
14207: Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single );
14208: Procedure glg1SetVector7( var v : TVectorGL; const vSrc : TVectorGL );
14209: Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single );
14210: Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector );
14211: Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL );
14212: Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single );
14213: Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single );
14214: Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector );
14215: Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL );
14216: Procedure glRstVector( var v : TAffineVector );
14217: Procedure glRstVector1( var v : TVectorGL );
14218: Function glVectorAdd( const v1, v2 : TAffineVector ) : TAffineVector;
14219: Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector );
14220: //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector );

```

```

14221: Function glVectorAdd3( const v1, v2 : TVectorGL ) : TVectorGL;
14222: Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL );
14223: Function glVectorAdd5( const v : TAffineVector; const f : Single ) : TAffineVector;
14224: Function glVectorAdd6( const v : TVectorGL; const f : Single ) : TVectorGL;
14225: Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector );
14226: Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL );
14227: Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL );
14228: Procedure glAddVector10( var v : TAffineVector; const f : Single );
14229: Procedure glAddVector11( var v : TVectorGL; const f : Single );
14230: //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Int;dest:PTexPointArray);
14231: //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Integer;const scale:TTexPoint; dest : PTExPointArray);
14232: //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
PAffineVectorArray);
14233: Function glVectorSubtract( const V1, V2 : TAffineVector ) : TAffineVector;
14234: Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector );
14235: Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL );
14236: Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL );
14237: Function glVectorSubtract4( const V1, V2 : TVectorGL ) : TVectorGL;
14238: Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL );
14239: Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector );
14240: Function glVectorSubtract7( const v1 : TAffineVector; delta : Single ) : TAffineVector;
14241: Function glVectorSubtract8( const v1 : TVectorGL; delta : Single ) : TVectorGL;
14242: Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector );
14243: Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL );
14244: Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single );
14245: //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat );
14246: Function glTexPointCombine( const t1, t2 : TTExPoint; f1, f2 : Single ) : TTExPoint;
14247: Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single ) : TAffineVector;
14248: Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single ) : TAffineVector;
14249: Procedure glVectorCombine34( const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector );
14250: Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single );
14251: Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single );
14252: Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single ) : TVectorGL;
14253: Function glVectorCombine8( const V1 : TVectorGL; const V2 : TAffineVector; const F1,F2:Single ) : TVectorGL;
14254: Procedure glVectorCombine9( const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL );
14255: Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL );
14256: Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL );
14257: Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single ) : TVectorGL;
14258: Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL );
14259: Function glVectorDotProduct( const V1, V2 : TAffineVector ) : Single;
14260: Function glVectorDotProduct1( const V1, V2 : TVectorGL ) : Single;
14261: Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector ) : Single;
14262: Function glPointProject( const p, origin, direction : TAffineVector ) : Single;
14263: Function glPointProject1( const p, origin, direction : TVectorGL ) : Single;
14264: Function glVectorCrossProduct( const V1, V2 : TAffineVector ) : TAffineVector;
14265: Function glVectorCrossProduct1( const V1, V2 : TVectorGL ) : TVectorGL;
14266: Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL );
14267: Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL );
14268: Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector );
14269: Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector );
14270: Function glLerp( const start, stop, t : Single ) : Single;
14271: Function glAngleLerp( start, stop, t : Single ) : Single;
14272: Function glDistanceBetweenAngles( angle1, angle2 : Single ) : Single;
14273: Function glTexPointLerp( const t1, t2 : TTExPoint; t : Single ) : TTExPoint;
14274: Function glVectorLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14275: Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector );
14276: Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single ) : TVectorGL;
14277: Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL );
14278: Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14279: Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single ) : TAffineVector;
14280: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray );
14281: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
PAffineVectorArray );
14282: Function glVectorLength( const x, y : Single ) : Single;
14283: Function glVectorLength1( const x, y, z : Single ) : Single;
14284: Function glVectorLength2( const v : TAffineVector ) : Single;
14285: Function glVectorLength3( const v : TVectorGL ) : Single;
14286: Function glVectorLength4( const v : array of Single ) : Single;
14287: Function glVectorNorm( const x, y : Single ) : Single;
14288: Function glVectorNorm1( const v : TAffineVector ) : Single;
14289: Function glVectorNorm2( const v : TVectorGL ) : Single;
14290: Function glVectorNorm3( var V : array of Single ) : Single;
14291: Procedure glNormalizeVector( var v : TAffineVector );
14292: Procedure glNormalizeVector1( var v : TVectorGL );
14293: Function glVectorNormalize( const v : TAffineVector ) : TAffineVector;
14294: Function glVectorNormalize1( const v : TVectorGL ) : TVectorGL;
14295: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer );
14296: Function glVectorAngleCosine( const V1, V2 : TAffineVector ) : Single;
14297: Function glVectorNegate( const v : TAffineVector ) : TAffineVector;
14298: Function glVectorNegate1( const v : TVectorGL ) : TVectorGL;
14299: Procedure glNegateVector( var V : TAffineVector );
14300: Procedure glNegateVector2( var V : TVectorGL );
14301: Procedure glNegateVector3( var V : array of Single );
14302: Procedure glScaleVector( var v : TAffineVector; factor : Single );
14303: Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector );
14304: Procedure glScaleVector2( var v : TVectorGL; factor : Single );
14305: Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL );

```

```

14306: Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14307: Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14308: Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14309: Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14310: Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14311: Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14312: Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14313: Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14314: Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14315: Function glVectorIsNull( const v : TVectorGL) : Boolean;
14316: Function glVectorIsNotNull( const v : TAffineVector) : Boolean;
14317: Function glVectorSpacing( const v1, v2 : TTExPoint) : Single;
14318: Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14319: Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14320: Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14321: Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14322: Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14323: Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14324: Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector;
14325: Function glVectorReflect( const V, N : TAffineVector) : TAffineVector;
14326: Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14327: Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14328: Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single);
14329: Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14330: Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14331: Procedure glVectorRotateAroundY1( const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14332: Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14333: Procedure glAbsVector( var v : TVectorGL);
14334: Procedure glAbsVector1( var v : TAffineVector);
14335: Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14336: Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14337: Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14338: Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14339: Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14340: Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14341: Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14342: Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14343: Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14344: Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14345: Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14346: Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14347: Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14348: Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14349: Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14350: Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14351: Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14352: Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14353: Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14354: Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix;
14355: Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14356: Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14357: Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14358: Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14359: Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14360: Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14361: Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14362: Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14363: Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14364: Procedure glAdjointMatrix( var M : TMatrixGL);
14365: Procedure glAdjointMatrix1( var M : TAffineMatrix);
14366: Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14367: Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14368: Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14369: Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14370: Procedure glNormalizeMatrix( var M : TMatrixGL);
14371: Procedure glTransposeMatrix( var M : TAffineMatrix);
14372: Procedure glTransposeMatrix1( var M : TMatrixGL);
14373: Procedure glInvertMatrix( var M : TMatrixGL);
14374: Procedure glInvertMatrix1( var M : TAffineMatrix);
14375: Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL;
14376: Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean;
14377: Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14378: Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14379: Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14380: Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14381: Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane);
14382: Procedure glNormalizePlane( var plane : THmgPlane);
14383: Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14384: Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14385: Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14386: Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14387: Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14388: Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14389: Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14390: Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14391: Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14392: Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector;
14393: Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single;
14394: Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector

```

```

14395: Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector ) : Single
14396: Procedure SgsegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
14397: Segment0Closest, Segment1Closest : TAffineVector )
14398: Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14399: TEulerOrder', (' eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14400: Function glQuaternionMake( const Imag : array of Single; Real : Single ) : TQuaternion
14401: Function glQuaternionConjugate( const Q : TQuaternion ) : TQuaternion
14402: Procedure glQuaternionMagnitude( const Q : TQuaternion ) : Single
14403: Procedure glNormalizeQuaternion( var Q : TQuaternion )
14404: Function glQuaternionFromPoints( const V1, V2 : TAffineVector ) : TQuaternion
14405: Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector )
14406: Function glQuaternionFromMatrix( const mat : TMatrixGL ) : TQuaternion
14407: Function glQuaternionToMatrix( quat : TQuaternion ) : TMatrixGL
14408: Function glQuaternionToAffineMatrix( quat : TQuaternion ) : TAffineMatrix
14409: Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector ) : TQuaternion
14410: Function glQuaternionFromRollPitchYaw( const r, p, y : Single ) : TQuaternion
14411: Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder ) : TQuaternion
14412: Function glQuaternionMultiply( const qL, qR : TQuaternion ) : TQuaternion
14413: Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single ) : TQuaternion;
14414: Function glQuaternionSlerpl( const source, dest : TQuaternion; const t : Single ) : TQuaternion;
14415: Function glLnXp1( X : Extended ) : Extended
14416: Function glLog10( X : Extended ) : Extended
14417: Function glLog2( X : Extended ) : Extended;
14418: Function glLogN( Base, X : Extended ) : Extended
14419: Function glIntPower( Base : Extended; Exponent : Integer ) : Extended
14420: Function glPower( const Base, Exponent : Single ) : Single;
14421: Function glPowerl( Base : Single; Exponent : Integer ) : Single;
14422: Function glDegToRad( const Degrees : Extended ) : Extended;
14423: Function glDegToRad1( const Degrees : Single ) : Single;
14424: Function glRadToDeg( const Radians : Extended ) : Extended;
14425: Function glRadToDeg1( const Radians : Single ) : Single;
14426: Function glNormalizeAngle( angle : Single ) : Single
14427: Function glNormalizeDegAngle( angle : Single ) : Single
14428: Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended );
14429: Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double );
14430: Procedure glSinCos( const Theta : Single; var Sin, Cos : Single );
14431: Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended );
14432: Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double );
14433: Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single );
14434: Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single )
14435: Function glArcCos( const X : Extended ) : Extended;
14436: Function glArcCos1( const x : Single ) : Single;
14437: Function glArcSin( const X : Extended ) : Extended;
14438: Function glArcSin1( const X : Single ) : Single;
14439: Function glArcTan2l( const Y, X : Extended ) : Extended;
14440: Function glArcTan21( const Y, X : Single ) : Single;
14441: Function glFastArcTan2( y, x : Single ) : Single
14442: Function glTan( const X : Extended ) : Extended;
14443: Function glTan1( const X : Single ) : Single;
14444: Function glCoTan( const X : Extended ) : Extended;
14445: Function glCoTan1( const X : Single ) : Single;
14446: Function glSinh( const x : Single ) : Single;
14447: Function glSinh1( const x : Double ) : Double;
14448: Function glCosh( const x : Single ) : Single;
14449: Function glCosh1( const x : Double ) : Double;
14450: Function glRSqrt( v : Single ) : Single
14451: Function glRLength( x, y : Single ) : Single
14452: Function glISqrt( i : Integer ) : Integer
14453: Function glILength( x, y : Integer ) : Integer;
14454: Function glILength1( x, y, z : Integer ) : Integer;
14455: Procedure glRegisterBasedExp
14456: Procedure glRandomPointOnSphere( var p : TAffineVector )
14457: Function glRoundInt( v : Single ) : Single;
14458: Function glRoundInt1( v : Extended ) : Extended;
14459: Function glTrunc( v : Single ) : Integer;
14460: Function glTrunc64( v : Extended ) : Int64;
14461: Function glInt( v : Single ) : Single;
14462: Function glInt1( v : Extended ) : Extended;
14463: Function glFrac( v : Single ) : Single;
14464: Function glFrac1( v : Extended ) : Extended;
14465: Function glRound( v : Single ) : Integer;
14466: Function glRound64( v : Single ) : Int64;
14467: Function glRound641( v : Extended ) : Int64;
14468: Function glTrunc( X : Extended ) : Int64
14469: Function glRound( X : Extended ) : Int64
14470: Function glFrac( X : Extended ) : Extended
14471: Function glCeil( v : Single ) : Integer;
14472: Function glCeil64( v : Extended ) : Int64;
14473: Function glFloor( v : Single ) : Integer;
14474: Function glFloor64( v : Extended ) : Int64;
14475: Function glScaleAndRound( i : Integer; var s : Single ) : Integer
14476: Function glSign( x : Single ) : Integer
14477: Function glIsInRange( const x, a, b : Single ) : Boolean;
14478: Function glIsInRangel( const x, a, b : Double ) : Boolean;
14479: Function glIsInCube( const p, d : TAffineVector ) : Boolean;
14480: Function glIsInCubel( const p, d : TVectorGL ) : Boolean;
14481: //Function MinFloat( values : PSingleArray; nbItems : Integer ) : Single;
14482: //Function MinFloat1( values : PDoubleArray; nbItems : Integer ) : Double;

```

```

14483: //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14484: Function glMinFloat3( const v1, v2 : Single) : Single;
14485: Function glMinFloat4( const v : array of Single) : Single;
14486: Function glMinFloat5( const v1, v2 : Double) : Double;
14487: Function glMinFloat6( const v1, v2 : Extended) : Extended;
14488: Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14489: Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14490: Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14491: //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14492: //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14493: //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14494: Function glMaxFloat2( const v : array of Single) : Single;
14495: Function glMaxFloat3( const v1, v2 : Single) : Single;
14496: Function glMaxFloat4( const v1, v2 : Double) : Double;
14497: Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14498: Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14499: Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14500: Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14501: Function glMinInteger9( const v1, v2 : Integer) : Integer;
14502: Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14503: Function glMaxInteger( const v1, v2 : Integer) : Integer;
14504: Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14505: Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14506: //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14507: Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14508: //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14509: //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14510: Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14511: //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14512: Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14513: //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14514: Function glMaxXYZComponent( const v : TVectorGL) : Single;
14515: Function glMaxXYZComponent1( const v : TAffineVector) : single;
14516: Function glMinXYZComponent( const v : TVectorGL) : Single;
14517: Function glMinXYZComponent1( const v : TAffineVector) : single;
14518: Function glMaxAbsXYZComponent( v : TVectorGL) : Single;
14519: Function glMinAbsXYZComponent( v : TVectorGL) : Single;
14520: Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14521: Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14522: Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14523: Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14524: Procedure glSortArrayAscending( var a : array of Extended);
14525: Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14526: Function glClampValue1( const aValue, aMin : Single) : Single;
14527: Function glGeometryOptimizationMode : String;
14528: Procedure glBeginFPUOnlySection;
14529: Procedure glEndFPUOnlySection;
14530: Function glConvertRotation( const Angles : TAffineVector) : TVectorGL;
14531: Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector;
14532: Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector;
14533: Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector;
14534: Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector;
14535: Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector;
14536: Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector;
14537: Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean;
14538: Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word);
14539: Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14540: Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14541: Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14542: Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14543: Function glRoll( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14544: Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14545: Function glRayCastMinDistToPoint( const rayStart, rayVector : TVectorGL;const point:TVectorGL):Single;
14546: Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14547: Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14548: Function glSphereVisibleRadius( distance, radius : Single) : Single;
14549: Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum;
14550: Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci : TRenderContextClippingInfo) : Boolean;
14551: Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci : TRenderContextClippingInfo) : Boolean;
14552: Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14553: Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const Frustum:TFrustum):Bool;
14554: Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL;
14555: Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL;
14556: Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL;
14557: Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix;
14558: Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL;
14559: 'cPI','Single').setExtended( 3.141592654);
14560: 'cPIdiv180','Single').setExtended( 0.017453292);
14561: 'c180divPI','Single').setExtended( 57.29577951);
14562: 'c2PI','Single').setExtended( 6.283185307);
14563: 'cPIdiv2','Single').setExtended( 1.570796326);
14564: 'cPIdiv4','Single').setExtended( 0.785398163);
14565: 'c3PIdiv4','Single').setExtended( 2.35619449);
14566: 'cInv2PI','Single').setExtended( 1 / 6.283185307);

```

```

14567: 'cInv360','Single').setExtended( 1 / 360);
14568: 'c180','Single').setExtended( 180);
14569: 'c360','Single').setExtended( 360);
14570: 'cOneHalf','Single').setExtended( 0.5);
14571: 'cLn10','Single').setExtended( 2.302585093);
14572: {'MinSingle','Extended').setExtended( 1.5e-45);
14573: 'MaxSingle','Extended').setExtended( 3.4e+38);
14574: 'MinDouble','Extended').setExtended( 5.0e-324);
14575: 'MaxDouble','Extended').setExtended( 1.7e+308);
14576: 'MinExtended','Extended').setExtended( 3.4e-4932);
14577: 'MaxExtended','Extended').setExtended( 1.1e+4932);
14578: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14579: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14580: end;
14581:
14582: procedure SIRегистер_GLVectorFileObjects(CL: TPSpascalCompiler);
14583: begin
14584:   AddClassN(FindClass('TOBJECT'), 'TMeshObjectList'
14585:   (FindClass('TOBJECT'), 'TFaceGroups
14586: TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14587: TMeshAutoCenterings', 'set of TMeshAutoCentering
14588: TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14589: SIRегистер_TBaseMeshObject(CL);
14590: (FindClass('TOBJECT'), 'TSkeletonFrameList
14591: TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14592: SIRегистер_TSkeletonFrame(CL);
14593: SIRегистер_TSkeletonFrameList(CL);
14594: (FindClass('TOBJECT'), 'TSkeleton
14595: (FindClass('TOBJECT'), 'TSkeletonBone
14596: SIRегистер_TSkeletonBoneList(CL);
14597: SIRегистер_TSkeletonRootBoneList(CL);
14598: SIRегистер_TSkeletonBone(CL);
14599: (FindClass('TOBJECT'), 'TSkeletonColliderList
14600: SIRегистер_TSkeletonCollider(CL);
14601: SIRегистер_TSkeletonColliderList(CL);
14602: (FindClass('TOBJECT'), 'TGLBaseMesh
14603: TBleendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14604:   +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14605:   +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14606:   +'QuaternionList; end
14607: SIRегистер_TSkeleton(CL);
14608: TMeshObjectRenderingOption', '( moroGroupByMaterial )
14609: TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14610: SIRегистер_TMeshObject(CL);
14611: SIRегистер_TMeshObjectList(CL);
14612: //TMeshObjectListClass', 'class of TMeshObjectList
14613: (FindClass('TOBJECT'), 'TMeshMorphTargetList
14614: SIRегистер_TMeshMorphTarget(CL);
14615: SIRегистер_TMorphableMeshObject(CL);
14616: SIRегистер_TMorphableMeshObject(CL);
14617: TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14618: //PVerticesBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14619: //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14620: TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14621: SIRегистер_TSkeletonMeshObject(CL);
14622: SIRегистер_TFaceGroup(CL);
14623: TFaceGroupMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14624:   +'atTriangles, fgmmTriangleFan, fgmmQuads )
14625: SIRегистер_TFGVertexIndexList(CL);
14626: SIRегистер_TFGVertexNormalTexIndexList(CL);
14627: SIRегистер_TFGIndexTexCoordList(CL);
14628: SIRегистер_TFaceGroups(CL);
14629: TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14630: SIRегистер_TVectorFile(CL);
14631: //TVectorFileClass', 'class of TVectorFile
14632: SIRегистер_TGLGLSMVectorFile(CL);
14633: SIRегистер_TGLBaseMesh(CL);
14634: SIRегистер_TGLFreeForm(CL);
14635: TGLActorOption', '( aoSkeletonNormalizeNormals )
14636: TGLActorOptions', 'set of TGLActorOption
14637: 'cDefaultGLActorOptions', 'LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14638: (FindClass('TOBJECT'), 'TGLActor
14639: TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14640: SIRегистер_TActorAnimation(CL);
14641: TActorAnimationName', 'String
14642: SIRегистер_TActorAnimations(CL);
14643: SIRегистер_TGLBaseAnimationController(CL);
14644: SIRегистер_TGLAnimationController(CL);
14645: TActorFrameInterpolation', '( afpNone, afpLinear )
14646: TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14647:   +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14648: SIRегистер_TGLActor(CL);
14649: SIRегистер_TVectorFileFormat(CL);
14650: SIRегистер_TVectorFileFormatsList(CL);
14651: (FindClass('TOBJECT'), 'EInvalidVectorFile
14652: Function GetVectorFileFormats : TVectorFileFormatsList
14653: Function VectorFileFormatsFilter : String
14654: Function VectorFileFormatsSaveFilter : String
14655: Function VectorFileFormatExtensionByIndex( index : Integer ) : String

```

```

14656: Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14657: Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14658: end;
14659:
14660: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14661: begin
14662:   'Class_DColorPropPage', 'TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14663:   'Class_DFontPropPage', 'TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14664:   'Class_DPicturePropPage', 'TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14665:   'Class_DStringPropPage', 'TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14666:   SIRegister_TOLEStream(CL);
14667:   (FindClass('TOBJECT'), TConnectionPoints
14668:    TConnectionKind', '( ckSingle, ckMulti )
14669:   SIRegister_TConnectionPoint(CL);
14670:   SIRegister_TConnectionPoints(CL);
14671:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14672:   (FindClass('TOBJECT'), TActiveXControlFactory
14673:   SIRegister_TActiveXControl(CL);
14674:   //TActiveXControlClass', 'class of TActiveXControl
14675:   SIRegister_TActiveXControlFactory(CL);
14676:   SIRegister_TActiveFormControl(CL);
14677:   SIRegister_TActiveForm(CL);
14678:   //TActiveFormClass', 'class of TActiveForm
14679:   SIRegister_TActiveFormFactory(CL);
14680:   (FindClass('TOBJECT'), TPropertyPageImpl
14681:   SIRegister_TPropertyPage(CL);
14682:   //TPROPERTYPAGECLASS', 'class of TPropertyPage
14683:   SIRegister_TPropertyPageImpl(CL);
14684:   SIRegister_TActiveXPropertyPage(CL);
14685:   SIRegister_TActiveXPropertyPageFactory(CL);
14686:   SIRegister_TCustomAdapter(CL);
14687:   SIRegister_TAdapterNotifier(CL);
14688:   SIRegister_IFontAccess(CL);
14689:   SIRegister_TFontAdapter(CL);
14690:   SIRegister_IPictureAccess(CL);
14691:   SIRegister_TPictureAdapter(CL);
14692:   SIRegister_TOLEGraphic(CL);
14693:   SIRegister_TStringsAdapter(CL);
14694:   SIRegister_TReflectorWindow(CL);
14695:   Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14696:   Procedure GetOLEFont( Font : TFont; var OleFont : IFontDisp)
14697:   Procedure SetOLEFont( Font : TFont; OleFont : IFontDisp)
14698:   Procedure GetOLEPicture( Picture : TPicture; var OlePicture : IPictureDisp)
14699:   Procedure SetOLEPicture( Picture : TPicture; OlePicture : IPictureDisp)
14700:   Procedure GetOLEStrings( Strings : TStrings; var OleStrings : IStrings)
14701:   Procedure SetOLEStrings( Strings : TStrings; OleStrings : IStrings)
14702:   Function ParkingWindow : HWND
14703: end;
14704:
14705: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14706: begin
14707:   // TIP6Bytes = array [0..15] of Byte;
14708:   {binary form of IPv6 adress (for string conversion routines)}
14709:   // TIP6Words = array [0..7] of Word;
14710:   AddTypeS('TIP6Bytes', 'array [0..15] of Byte;');
14711:   AddTypeS('TIP6Words', 'array [0..7] of Word;');
14712:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4 : Byte; end');
14713:   Function synaIsIP( const Value : string) : Boolean';
14714:   Function synaIsIP6( const Value : string) : Boolean';
14715:   Function synaPToID( Host : string) : Ansistring';
14716:   Function synaStrToIp6( value : string) : TIP6Bytes';
14717:   Function synaIp6ToStr( value : TIP6Bytes) : string';
14718:   Function synaStrToIp( value : string) : integer';
14719:   Function synaIpToStr( value : integer) : string';
14720:   Function synaReverseIP( Value : AnsiString) : AnsiString';
14721:   Function synaReverseIP6( Value : AnsiString) : AnsiString';
14722:   Function synaExpandIP6( Value : AnsiString) : AnsiString';
14723:   Function xStrToIP( const Value : string) : TIPAdr';
14724:   Function xIPToStr( const Adresse : TIPAdr) : string';
14725:   Function IPToCardinal( const Adresse : TIPAdr) : Cardinal';
14726:   Function CardinalToIP( const Value : Cardinal) : TIPAdr';
14727: end;
14728:
14729: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14730: begin
14731:   AddTypeS('TSpecials', 'set of Char');
14732:   Const ('SpecialChar', 'TSpecials').SetSet( '='[]<>;@/?\-' );
14733:   Const ('URLFullSpecialChar', 'TSpecials').SetSet( '/?:@=&#+' );
14734:   Const ('TableBase64'ABCDEFIGHJKLMNOPQRSTUVWXYZZabcdefg hijklmnopqrstuvwxyz0123456789+/=+');
14735:   Const ('TableBase64mod'ABCDEFIGHJKLMNOPQRSTUVWXYZZabcdefg hijklmnopqrstuvwxyz0123456789+,=+');
14736:   Const ('TableUU'(`!#$%`)*+,-./0123456789:+<>?@ABCDEFIGHJKLMNOPQRSTUVWXYZ[\]^_');
14737:   Const ('TableXX'(+0123456789ABCDEFIGHJKLMNOPQRSTUVWXYZZabcdefg hijklmnopqrstuvwxyz');
14738:   Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString';
14739:   Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString';
14740:   Function DecodeURL( const Value : AnsiString) : AnsiString';
14741:   Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString';
14742:   Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString';
14743:   Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString';
14744:   Function EncodeURLElement( const Value : AnsiString) : AnsiString';

```

```

14745: Function EncodeURL( const Value : AnsiString ) : AnsiString');
14746: Function Decode4to3( const Value, Table : AnsiString ) : AnsiString');
14747: Function Decode4to3Ex( const Value, Table : AnsiString ) : AnsiString');
14748: Function Encode3to4( const Value, Table : AnsiString ) : AnsiString');
14749: Function synDecodeBase64( const Value : AnsiString ) : AnsiString');
14750: Function synEncodeBase64( const Value : AnsiString ) : AnsiString');
14751: Function DecodeBase64mod( const Value : AnsiString ) : AnsiString');
14752: Function EncodeBase64mod( const Value : AnsiString ) : AnsiString');
14753: Function DecodeUU( const Value : AnsiString ) : AnsiString');
14754: Function EncodeUU( const Value : AnsiString ) : AnsiString');
14755: Function DecodeXX( const Value : AnsiString ) : AnsiString');
14756: Function DecodeYEnc( const Value : AnsiString ) : AnsiString');
14757: Function UpdateCrc32( Value : Byte; Crc32 : Integer ) : Integer');
14758: Function synCrc32( const Value : AnsiString ) : Integer');
14759: Function UpdateCrc16( Value : Byte; Crc16 : Word ) : Word');
14760: Function Crc16( const Value : AnsiString ) : Word');
14761: Function synMD5( const Value : AnsiString ) : AnsiString');
14762: Function HMAC_MD5( Text, Key : AnsiString ) : AnsiString');
14763: Function MD5LongHash( const Value : AnsiString; Len : integer ) : AnsiString');
14764: Function synSHA1( const Value : AnsiString ) : AnsiString');
14765: Function HMAC_SHA1( Text, Key : AnsiString ) : AnsiString');
14766: Function SHA1LongHash( const Value : AnsiString; Len : integer ) : AnsiString');
14767: Function synMD4( const Value : AnsiString ) : AnsiString');
14768: end;
14769:
14770: procedure SIRegister_synamachar(CL: TPSPPascalCompiler);
14771: begin
14772:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14773:             +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14774:             +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14775:             +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14776:             +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE, '
14777:             +'C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14778:             +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14779:             +', NEXTSTEP, ARMSCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14780:             +'IS620, CP874, VISCN, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14781:             +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14782:             +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14783:             +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14784:             +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14785:             +', CP864, CP865, CP869, CP1125 ')');
14786:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14787:   Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14788:   Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
14789:     TransformTable : array of Word) : AnsiString );
14790:   Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
14791:     TransformTable : array of Word; Translit : Boolean) : AnsiString );
14792:   Function GetCurCP : TMimeChar');
14793:   Function GetCurOEMCP : TMimeChar');
14794:   Function GetCFFromID( Value : AnsiString ) : TMimeChar');
14795:   Function GetIDFromCP( Value : TMimeChar ) : AnsiString );
14796:   Function NeedCharsetConversion( const Value : AnsiString ) : Boolean );
14797:   Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14798:   Function GetBOM( Value : TMimeChar ) : AnsiString );
14799:   Function StringToWide( const Value : AnsiString ) : WideString );
14800:   Function WideToString( const Value : WideString ) : AnsiString );
14801: end;
14802: procedure SIRegister_synamisc(CL: TPSPPascalCompiler);
14803: begin
14804:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14805:   Procedure WakeOnLan( MAC, IP : string );
14806:   Function GetDNS : string );
14807:   Function GetIEProxy( protocol : string ) : TProxySetting );
14808:   Function GetLocalIPs : string );
14809: end;
14810:
14811: procedure SIRegister_synaser(CL: TPSPPascalCompiler);
14812: begin
14813:   AddConstantN('synCR','Char #$0d');
14814:   Const('synLF','Char #$0a');
14815:   Const('cSerialChunk','LongInt'( 8192));
14816:   Const('LockfileDirectory','String '/var/lock');
14817:   Const('PortIsClosed','LongInt'( - 1);
14818:   Const('ErrAlreadyOwned','LongInt'( 9991);
14819:   Const('ErrAlreadyInUse','LongInt'( 9992);
14820:   Const('ErrWrongParameter','LongInt'( 9993);
14821:   Const('ErrPortNotOpen','LongInt'( 9994);
14822:   Const('ErrNoDeviceAnswer','LongInt'( 9995);
14823:   Const('ErrMaxBuffer','LongInt'( 9996);
14824:   Const('ErrTimeout','LongInt'( 9997);
14825:   Const('ErrNotRead','LongInt'( 9998);
14826:   Const('ErrFrame','LongInt'( 9999);
14827:   Const('ErrOverrun','LongInt'( 10000);
14828:   Const('ErrRxOver','LongInt'( 10001);
14829:   Const('ErrRxParity','LongInt'( 10002);
14830:   Const('ErrTxFull','LongInt'( 10003);
14831:   Const('dcb_Binary','LongWord')( $00000001);

```

```

14832: Const('dcb_ParityCheck','LongWord')( $00000002);
14833: Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14834: Const('dcb_OutxCsrFlow','LongWord')( $00000008);
14835: Const('dcb_DtrControlMask','LongWord')( $00000030);
14836: Const('dcb_DtrControlDisable','LongWord')( $00000000);
14837: Const('dcb_DtrControlEnable','LongWord')( $00000010);
14838: Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14839: Const('dcb_DsrSensitivity','LongWord')( $00000040);
14840: Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14841: Const('dcb_OutX','LongWord')( $00000100);
14842: Const('dcb_InX','LongWord')( $00000200);
14843: Const('dcb_ErrorChar','LongWord')( $00000400);
14844: Const('dcb_NullStrip','LongWord')( $00000800);
14845: Const('dcb_RtsControlMask','LongWord')( $00003000);
14846: Const('dcb_RtsControlDisable','LongWord')( $00000000);
14847: Const('dcb_RtsControlEnable','LongWord')( $00001000);
14848: Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14849: Const('dcb_RtsControlToggle','LongWord')( $00003000);
14850: Const('dcb_AbortOnError','LongWord')( $00004000);
14851: Const('dcb_Reservesd','LongWord')( $FFFF8000);
14852: Const('synSBL','LongInt'( 0));
14853: Const('SBlandHalf','LongInt'( 1));
14854: Const('synSB2','LongInt'( 2));
14855: Const('synINVALID_HANDLE_VALUE','LongInt'( THandle( - 1 )));
14856: Const('CS7fix','LongWord')( $0000020);
14857: AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long' +
14858:   + 'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par' +
14859:   + 'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : ' +
14860:   + 'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14861: //AddTypeS('PDCB', '^TDCB // will not work');
14862: //Const('MaxRates','LongInt'( 18));
14863: //Const('MaxRates','LongInt'( 30));
14864: //Const('MaxRates','LongInt'( 19));
14865: Const('O_SYNC','LongWord')( $0080);
14866: Const('synOK','LongInt'( 0));
14867: Const('synErr','LongInt'( integer( - 1 )));
14868: AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,' +
14869:   HR_WriteCount, HR_Wait )');
14870: Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string'));
14871: SIRegister_ESynaSerError(CL);
14872: SIRegister_TBlockSerial(CL);
14873: Function GetSerialPortNames : string);
14874: end;
14875: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14876: begin
14877: Const('DLLIconvName','String 'libiconv.so');
14878: Const('DLLIconvName','String 'iconv.dll');
14879: AddTypeS('size_t','Cardinal');
14880: AddTypeS('iconv_t','Integer');
14881: //AddTypeS('iconv_t','Pointer');
14882: AddTypeS('argptr','iconv_t');
14883: Function SynalconvOpen( const tocode, fromcode : Ansistring) : iconv_t';
14884: Function SynalconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t';
14885: Function SynalconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t';
14886: Function Synalconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer';
14887: Function SynalconvClose( var cd : iconv_t) : integer';
14888: Function SynalconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer';
14889: Function IsIconvloaded : Boolean';
14890: Function InitIconvInterface : Boolean';
14891: Function DestroyIconvInterface : Boolean';
14892: Const('ICONV_TRIVIALP','LongInt'( 0));
14893: Const('ICONV_GET_TRANSLITERATE','LongInt'( 1));
14894: Const('ICONV_SET_TRANSLITERATE','LongInt'( 2));
14895: Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3));
14896: Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4));
14897: end;
14898:
14899: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14900: begin
14901: Const 'ICMP_ECHO','LongInt'( 8);
14902: Const('ICMP_ECHOREPLY','LongInt'( 0);
14903: Const('ICMP_UNREACH','LongInt'( 3);
14904: Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14905: Const('ICMP6_ECHO','LongInt'( 128);
14906: Const('ICMP6_ECHOREPLY','LongInt'( 129);
14907: Const('ICMP6_UNREACH','LongInt'( 1);
14908: Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14909: AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOr' +
14910:   + 'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14911: SIRegister_TPINGSend(CL);
14912: Function PingHost( const Host : string) : Integer';
14913: Function TraceRouteHost( const Host : string) : string';
14914: end;
14915:
14916: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14917: begin
14918: AddConstantN('synASN1_BOOL','LongWord')( $01);
14919: Const('synASN1_INT','LongWord')( $02);

```

```

14920: Const('synASN1_OCTSTR','LongWord')( $04);
14921: Const('synASN1_NULL','LongWord')( $05);
14922: Const('synASN1_OBJID','LongWord')( $06);
14923: Const('synASN1_ENUM','LongWord')( $0a);
14924: Const('synASN1_SEQ','LongWord')( $30);
14925: Const('synASN1_SETOF','LongWord')( $31);
14926: Const('synASN1_IPADDR','LongWord')( $40);
14927: Const('synASN1_COUNTER','LongWord')( $41);
14928: Const('synASN1_GAUGE','LongWord')( $42);
14929: Const('synASN1_TIMETICKS','LongWord')( $43);
14930: Const('synASN1_OPAQUE','LongWord')( $44);
14931: Function synASNEncOIDItem( Value : Integer ) : AnsiString');
14932: Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString ) : Integer');
14933: Function synASNEncLen( Len : Integer ) : AnsiString');
14934: Function synASNDecLen( var Start : Integer; const Buffer : AnsiString ) : Integer');
14935: Function synASNEncInt( Value : Integer ) : AnsiString');
14936: Function synASNEncUInt( Value : Integer ) : AnsiString');
14937: Function synASNObject( const Data : AnsiString; ASNType : Integer ) : AnsiString');
14938: Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14939: Function synMibToId( Mib : String ) : AnsiString';
14940: Function synIdToMib( const Id : AnsiString ) : String';
14941: Function synIntMibToStr( const Value : AnsiString ) : AnsiString');
14942: Function ASNdump( const Value : AnsiString ) : AnsiString');
14943: Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings ) : Boolean');
14944: Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString');
14945: end;
14946:
14947: procedure SIRegister_ldapsend(CL: TPSCompiler);
14948: begin
14949: Const('cLDAPProtocol','String '389');
14950: Const('LDAP ASN1 BIND REQUEST','LongWord')( $60);
14951: Const('LDAP ASN1 BIND RESPONSE','LongWord')( $61);
14952: Const('LDAP ASN1 UNBIND REQUEST','LongWord')( $42);
14953: Const('LDAP ASN1 SEARCH REQUEST','LongWord')( $63);
14954: Const('LDAP ASN1 SEARCH ENTRY','LongWord')( $64);
14955: Const('LDAP ASN1 SEARCH DONE','LongWord')( $65);
14956: Const('LDAP ASN1 SEARCH REFERENCE','LongWord')( $73);
14957: Const('LDAP ASN1 MODIFY REQUEST','LongWord')( $66);
14958: Const('LDAP ASN1 MODIFY RESPONSE','LongWord')( $67);
14959: Const('LDAP ASN1 ADD REQUEST','LongWord')( $68);
14960: Const('LDAP ASN1 ADD RESPONSE','LongWord')( $69);
14961: Const('LDAP ASN1 DEL REQUEST','LongWord')( $4A);
14962: Const('LDAP ASN1 DEL RESPONSE','LongWord')( $6B);
14963: Const('LDAP ASN1 MODIFYDN REQUEST','LongWord')( $6C);
14964: Const('LDAP ASN1 MODIFYDN RESPONSE','LongWord')( $6D);
14965: Const('LDAP ASN1 COMPARE REQUEST','LongWord')( $5E);
14966: Const('LDAP ASN1 COMPARE RESPONSE','LongWord')( $6F);
14967: Const('LDAP ASN1 ABANDON REQUEST','LongWord')( $70);
14968: Const('LDAP ASN1 EXT REQUEST','LongWord')( $77);
14969: Const('LDAP ASN1 EXT RESPONSE','LongWord')( $78);
14970: SIRegister_TLDAPAttribute(CL);
14971: SIRegister_TLDAPAttributeList(CL);
14972: SIRegister_TLDAPResult(CL);
14973: SIRegister_TLDAPResultList(CL);
14974: AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14975: AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14976: AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14977: SIRegister_TLDAPSnd(CL);
14978: Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString');
14979: end;
14980:
14981:
14982: procedure SIRegister_slogsend(CL: TPSCompiler);
14983: begin
14984: Const('cSysLogProtocol','String '514');
14985: Const('FCL_Kernel','LongInt'( 0));
14986: Const('FCL_UserLevel','LongInt'( 1));
14987: Const('FCL_MailSystem','LongInt'( 2));
14988: Const('FCL_System','LongInt'( 3));
14989: Const('FCL_Security','LongInt'( 4));
14990: Const('FCL_Syslogd','LongInt'( 5));
14991: Const('FCL_Printer','LongInt'( 6));
14992: Const('FCL_News','LongInt'( 7));
14993: Const('FCL_UUCP','LongInt'( 8));
14994: Const('FCL_Clock','LongInt'( 9));
14995: Const('FCL_Authorization','LongInt'( 10));
14996: Const('FCL_FTP','LongInt'( 11));
14997: Const('FCL_NTP','LongInt'( 12));
14998: Const('FCL_LogAudit','LongInt'( 13));
14999: Const('FCL_LogAlert','LongInt'( 14));
15000: Const('FCL_Time','LongInt'( 15));
15001: Const('FCL_Local0','LongInt'( 16));
15002: Const('FCL_Local1','LongInt'( 17));
15003: Const('FCL_Local2','LongInt'( 18));
15004: Const('FCL_Local3','LongInt'( 19));
15005: Const('FCL_Local4','LongInt'( 20));
15006: Const('FCL_Local5','LongInt'( 21));
15007: Const('FCL_Local6','LongInt'( 22));
15008: Const('FCL_Local7','LongInt'( 23));

```

```

15009: Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
15010: SIRegister_TSyslogMessage(CL);
15011: SIRegister_TSyslogSend(CL);
15012: Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
Content:string):Boolean;
15013: end;
15014:
15015:
15016: procedure SIRegister_mimemess(CL: TPSCompiler);
15017: begin
15018:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
15019:   SIRegister_TMessHeader(CL);
15020: //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
15021:   SIRegister_TMimeMess(CL);
15022: end;
15023:
15024: procedure SIRegister_mimepart(CL: TPSCompiler);
15025: begin
15026:   (FindClass('TOBJECT'),'TMimePart');
15027:   AddTypes('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
15028:   AddTypes('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
15029:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
15030:   SIRegister_TMimePart(CL);
15031: Const('MaxMineType','LongInt'( 25));
15032: Function GenerateBoundary : string';
15033: end;
15034:
15035: procedure SIRegister_mimeinln(CL: TPSCompiler);
15036: begin
15037:   Function InlineDecode( const Value : string; CP : TMimeChar) : string';
15038:   Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string';
15039:   Function NeedInline( const Value : AnsiString) : boolean';
15040:   Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
15041:   Function InlineCode( const Value : string) : string');
15042:   Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
15043:   Function InlineEmail( const Value : string) : string');
15044: end;
15045:
15046: procedure SIRegister_ftpsend(CL: TPSCompiler);
15047: begin
15048: Const('cFtpProtocol','String '21');
15049: Const('cFtpDataProtocol','String '20');
15050: Const('FTP_OK','LongInt'( 255);
15051: Const('FTP_ERR','LongInt'( 254);
15052: AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15053: SIRegister_TFTPLListRec(CL);
15054: SIRegister_TFTPLList(CL);
15055: SIRegister_TFTPSend(CL);
15056: Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean';
15057: Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean';
15058: Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
ToPort, ToFile, ToUser, ToPass : string) : Boolean';
15059: end;
15060:
15061: procedure SIRegister_httpsend(CL: TPSCompiler);
15062: begin
15063: Const('cHttpProtocol','String '80');
15064: AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15065: SIRegister_THTTFSend(CL);
15066: Function HttpGetText( const URL : string; const Response : TStrings) : Boolean';
15067: Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean';
15068: Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean';
15069: Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean';
15070: Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
ResultData:TStrings):Bool;
15071: end;
15072:
15073: procedure SIRegister_smtpsend(CL: TPSCompiler);
15074: begin
15075: Const('cSmtpProtocol','String '25');
15076: SIRegister_TSMTSPSend(CL);
15077: Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
Passw:string):Bool;
15078: Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15079: Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
Usrname, Password : string):Boolean';
15080: end;
15081:
15082: procedure SIRegister_snmpsend(CL: TPSCompiler);
15083: begin
15084: Const('cSnmpProtocol','String '161');
15085: Const('cSnmpTrapProtocol','String '162');
15086: Const('SNMP_V1','LongInt'( 0);
15087: Const('SNMP_V2C','LongInt'( 1);
15088: Const('SNMP_V3','LongInt'( 3);
15089: Const('PDUGetRequest','LongWord')($A0);
15090: Const('PDUGetNextRequest','LongWord')($A1);
15091: Const('PDUGetResponse','LongWord')($A2);
15092: Const('PDUSetRequest','LongWord')($A3);

```

```

15093: Const('PDUTrap','LongWord')($A4);
15094: Const('PDUGetBulkRequest','LongWord')($A5);
15095: Const('PDUInformRequest','LongWord')($A6);
15096: Const('PDUTrapV2','LongWord')($A7);
15097: Const('PDUReport','LongWord')($A8);
15098: Const('ENoError',LongInt 0;
15099: Const('ETooBig','LongInt')( 1);
15100: Const('ENoSuchName','LongInt'( 2);
15101: Const('EBadValue','LongInt'( 3);
15102: Const('EReadOnly','LongInt'( 4);
15103: Const('EGenErr','LongInt'( 5);
15104: Const('ENOAccess','LongInt'( 6);
15105: Const('EWrongType','LongInt'( 7);
15106: Const('EWrongLength','LongInt'( 8);
15107: Const('EWrongEncoding','LongInt'( 9);
15108: Const('EWrongValue','LongInt'( 10);
15109: Const('ENOCreation','LongInt'( 11);
15110: Const('EInconsistentValue','LongInt'( 12);
15111: Const('EResourceUnavailable','LongInt'( 13);
15112: Const('ECommitFailed','LongInt'( 14);
15113: Const('EUndoFailed','LongInt'( 15);
15114: Const('EAuthorizationError','LongInt'( 16);
15115: Const('ENotWritable','LongInt'( 17);
15116: Const('EInconsistentName','LongInt'( 18);
15117: AddTypes('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15118: AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15119: AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15120: SIRegister_TSNSMPMib(CL);
15121: AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer;
15122:   +EngineTime : integer; EngineStamp : Cardinal; end');
15123: SIRegister_TSNSMPRec(CL);
15124: SIRegister_TSNSMPSend(CL);
15125: Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString ) : Boolean';
15126: Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer ) : Boolean';
15127: Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15128: Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings) : Boolean;
15129: Function SNMPGetTableElement(const BaseOID,RowID,Colid,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15130: Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds : Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer ) : Integer';
15131: Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds : Integer; const MIBName, MIBValue : TStringList ) : Integer';
15132: end;
15133:
15134: procedure SIRegister_NetWork(CL: TPPascalCompiler);
15135: begin
15136:   Function GetDomainName2: AnsiString';
15137:   Function GetDomainController( Domain : AnsiString ) : AnsiString';
15138:   Function GetDomainUsers( Controller : AnsiString ) : AnsiString';
15139:   Function GetDomainGroups( Controller : AnsiString ) : AnsiString';
15140:   Function GetDateTIme( Controller : AnsiString ) : TDateTime';
15141:   Function GetFullName2( Controller, UserID : AnsiString ) : AnsiString';
15142: end;
15143:
15144: procedure SIRegister_wwSystem(CL: TPPascalCompiler);
15145: begin
15146:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15147:   TwwDateTimeSelection', '(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM)';
15148:   Function wwStrToDate( const S : string ) : boolean';
15149:   Function wwStrToTime( const S : string ) : boolean';
15150:   Function wwStrToDateTIme( const S : string ) : boolean';
15151:   Function wwStrToTimeVal( const S : string ) : TDateTime';
15152:   Function wwStrToDateVal( const S : string ) : TDateTime';
15153:   Function wwStrToDateTImeVal( const S : string ) : TDateTime';
15154:   Function wwStrToInt( const S : string ) : boolean';
15155:   Function wwStrToFloat( const S : string ) : boolean';
15156:   Function wwGetDateOrder( const DateFormat : string ) : TwwDateOrder';
15157:   Function wwNextDay( Year, Month, Day : Word ) : integer';
15158:   Function wwPriorDay( Year, Month, Day : Word ) : integer';
15159:   Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime ) : Boolean';
15160:   Function wwDoDecodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime ) : Boolean';
15161:   Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool:TwwDateTimeSelection');
15162:   Function wwGetTimeCursorPosition( SelStart : integer; Text : string ) : TwwDateTimeSelection';
15163:   Function wwScanDate( const S : string; var Date : TDateTime ) : Boolean';
15164:   Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer ) : Boolean';
15165:   Procedure wwSetDateTimeCursorPosition(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15166:   Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean';
15167: end;
15168:
15169: unit uPSI_Themes;
15170: Function ThemeServices : TThemeServices';
15171: Function ThemeControl( AControl : TControl ) : Boolean';
15172: Function UnthemedDesigner( AControl : TControl ) : Boolean';
15173: procedure SIRegister_UDDIHelper(CL: TPPascalCompiler);
15174: begin
15175:   Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String';
15176: end;
15177: Unit upSC_menus;
15178:   Function StripHotkey( const Text : string ) : string';
15179:   Function GetHotkey( const Text : string ) : string';

```

```

15180: Function AnsiSameCaption( const Text1, Text2 : string ) : Boolean';
15181: Function IsAltGRRPressed : boolean';
15182:
15183: procedure SIRегистер_IdIMAP4Server(CL: TPSCompiler);
15184: begin
15185:   TCommandEvent', 'Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean)';
15186:   SIRегистер_TIdIMAP4Server(CL);
15187: end;
15188:
15189: procedure SIRегистер_VariantSymbolTable(CL: TPSCompiler);
15190: begin
15191:   'HASH_SIZE','LongInt'( 256 );
15192:   CL.FindClass('TOBJECT','EVariantSymbolTable');
15193:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15194:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15195:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15196:     + ' : Integer; Value : Variant; end');
15197:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15198:   SIRегистер_TVariantSymbolTable(CL);
15199: end;
15200:
15201: procedure SIRегистер_udf_glob(CL: TPSCompiler);
15202: begin
15203:   SIRегистер_ThreadLocalVariables(CL);
15204:   Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD ) : PChar';
15205:   //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD ) : PISC_QUAD';
15206:   Function ThreadLocals : TThreadLocalVariables';
15207:   Procedure WriteDebug( sz : String );
15208:   CL.AddConstantN('UDF_SUCCESS','LongInt'( 0 );
15209:   'UDF_FAILURE','LongInt'( 1 );
15210:   'cSignificantlyLarger','LongInt'( 1024 * 4 );
15211:   CL.AddTypeS('mTByteArray', 'array of byte;');
15212:   function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15213:   function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15214:   procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15215:   function IsNetworkConnected: Boolean;
15216:   function IsInternetConnected: Boolean;
15217:   function IsCOMConnected: Boolean;
15218:   function IsNetworkOn: Boolean;
15219:   function IsInternetOn: Boolean;
15220:   function IsCOMON: Boolean;
15221:   Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15222:   TmrProc , 'procedure TmrProc(hWnd: HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);';
15223:   Function SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc ) : UINT';
15224:   Function KillTimer( hWnd : HWND; uIDEvent : UINT ) : BOOL';
15225:   Function wIsWindowUnicode( hWnd : HWND ) : BOOL';
15226:   Function wEnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL';
15227:   Function wIsWindowEnabled( hWnd : HWND ) : BOOL';
15228:   Function GetMenu( hWnd : HWND ) : HMENU';
15229:   Function SetMenu( hWnd : HWND; hMenu : HMENU ) : BOOL';
15230: end;
15231:
15232: procedure SIRегистер_SockTransport(CL: TPSCompiler);
15233: begin
15234:   SIRегистер_IDataBlock(CL);
15235:   SIRегистер_ISendDataBlock(CL);
15236:   SIRегистер_ITransport(CL);
15237:   SIRегистер_TDataBlock(CL);
15238:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15239:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15240:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15241:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15242:   SIRегистер_TCustomDataBlockInterpreter(CL);
15243:   SIRегистер_TSendDataBlock(CL);
15244:   'CallSig','LongWord')($D800);
15245:   'ResultsSig','LongWord')($D400);
15246:   'asMask','LongWord')($00FF);
15247:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EInterpreterError');
15248:   CL.AddClassN(CL.FindClass('TOBJECT'), 'ESocketConnectionError');
15249:   Procedure CheckSignature( Sig : Integer );
15250: end;
15251:
15252: procedure SIRегистер_WinInet(CL: TPSCompiler);
15253: begin
15254:   //CL.AddTypeS('HINTERNET', '__Pointer');
15255:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15256:   CL.AddTypeS('HINTERNET', 'Integer');
15257:   CL.AddTypeS('HINTERNET2', '__Pointer');
15258:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15259:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15260:   CL.AddTypeS('INTERNET_PORT', 'Word');
15261:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15262:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15263:   Function InternetTimeFromSystemTime(const pst: TSystemTime; dwRFC: DWORD; lpszTime: LPSTR; cbTime: DWORD) : BOOL;
15264:   'INTERNET_RFC1123_FORMAT','LongInt'( 0 );
15265:   'INTERNET_RFC1123_BUFSIZE','LongInt'( 30 );
15266:   Function InternetCrackUrl(lpszUrl1:PChar;dwUrlLength,dwFlags:DWORD;var
15267:   lpUrlComponents:TURLComponents):BOOL;
15268:   Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl1:PChar;var
15269:   dwUrlLength:DWORD):BOOL;

```

```

15268: Function InternetCloseHandle(hInet : HINTERNET) : BOOL';
15269: Function
InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15270: Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15271: ;dwContext:DWORD):HINTERNET;
15272: Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15273: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
dwContext:DWORD):BOOL;
15274: Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE ) : BOOL';
15275: Function
InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15276: Function InternetHangUp(dwConnection : DWORD; dwReserved : DWORD) : DWORD';
15277: Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD ) : BOOL';
15278: Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15279: Function InternetAutodialHangup( dwReserved : DWORD ) : BOOL';
15280: Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15281: Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15282: Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET ) : BOOL');
15283: Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15284: ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15285: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
dwContext:DWORD):BOOL;
15286: Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
TWin32FindData; dwFlags : DWORD; dwContext : DWORD ) : HINTERNET';
15287: Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD ) : BOOL';
15288: Function
WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15289: Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar ) : BOOL';
15290: Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15291: Function
FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15292: Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar ) : BOOL';
15293: Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar ) : BOOL';
15294: Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar ) : BOOL';
15295: Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15296: Function
FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15297: Function IS_GOPHER_FILE( GopherType : DWORD ) : BOOL';
15298: Function IS_GOPHER_DIRECTORY( GopherType : DWORD ) : BOOL';
15299: Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD ) : BOOL';
15300: Function IS_GOPHER_ERROR( GopherType : DWORD ) : BOOL';
15301: Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD ) : BOOL';
15302: Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD ) : BOOL';
15303: Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD ) : BOOL';
15304: Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD ) : BOOL';
15305: Function IS_GOPHER_ASK( GopherType : DWORD ) : BOOL';
15306: Function IS_GOPHER_PLUS( GopherType : DWORD ) : BOOL';
15307: Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD ) : BOOL';
15308: Function
GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
PChar;dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15309: Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL';
15310: Function
GopherOpenFile(hConec:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15311: Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15312: Function
HttpAddRequestHeaders(hReg:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15313: Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15314: Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15315: Function InternetAttemptConnect( dwReserved : DWORD ) : DWORD';
15316: Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15317: Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15318: Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LESTR; bPost : BOOL ) : DWORD';
15319: Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject ) : Int64';
15320: Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject ) : Bool';
15321: Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15322: Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15323: Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15324: Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15325: Function
InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15326: Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD ) : BOOL';
15327: end;
15328:
15329: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15330: begin
15331:   AddTypeS('strCharSet', 'set of char');
15332:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords );
15333:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15334:   Procedure strBreakApart( s : string; delimiter : string; parts : TStrings)');

```

```

15335: Function strGetToken( s : string; delimiter : string; var APos : integer ) : string');
15336: Procedure strStripPreceding( var s : string; delimiter : strCharSet') );
15337: Procedure strStripTrailing( var s : string; delimiter : strCharSet') );
15338: Procedure strStripWhiteSpace( var s : string') );
15339: Function strRemoveChar( str : string; removeChar : char ) : string');
15340: Function wwstrReplaceChar( str : string; removeChar, replaceChar : char ) : string');
15341: Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15342: Function wwEqualStr( s1, s2 : string) : boolean');
15343: Function strCount( s : string; delimiter : char) : integer');
15344: Function strWhiteSpace : strCharSet');
15345: Function wwExtractFileNameOnly( const FileName : string) : string');
15346: Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15347: Function strTrailing( s : string; delimiter : char ) : string');
15348: Function strPreceding( s : string; delimiter : char ) : string');
15349: Function wwstrReplace( s, Find, Replace : string ) : string');
15350: end;
15351:
15352: procedure SIRegister_DataBkr(CL: TPSPPascalCompiler);
15353: begin
15354:   SIRegister_TRemoteDataModule(CL);
15355:   SIRegister_TCRemoteDataModule(CL);
15356:   Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)' );
15357:   Procedure UnregisterPooled( const ClassID : string)' );
15358:   Procedure EnableSocketTransport( const ClassID : string)' );
15359:   Procedure DisableSocketTransport( const ClassID : string)' );
15360:   Procedure EnableWebTransport( const ClassID : string)' );
15361:   Procedure DisableWebTransport( const ClassID : string)' );
15362: end;
15363:
15364: procedure SIRegister_Mathbox(CL: TPSPPascalCompiler);
15365: begin
15366:   Function mxArcCos( x : Real ) : Real');
15367:   Function mxArcSin( x : Real ) : Real');
15368:   Function Comp2Str( N : Comp ) : String');
15369:   Function Int2StrPad0( N : LongInt; Len : Integer ) : String');
15370:   Function Int2Str( N : LongInt ) : String');
15371:   Function mxIsEqual( R1, R2 : Double ) : Boolean');
15372:   Function LogXY( x, y : Real ) : Real');
15373:   Function Pennies2Dollars( C : Comp ) : String');
15374:   Function mxPower( X : Integer; Y : Integer ) : Real');
15375:   Function Real2Str( N : Real; Width, Places : integer ) : String');
15376:   Function mxStr2Comp( MyString : string ) : Comp');
15377:   Function mxStr2Pennies( S : String ) : Comp');
15378:   Function Str2Real( MyString : string ) : Real');
15379:   Function XToTheY( x, y : Real ) : Real');
15380: end;
15381:
15382: //*****Cindy Functions!*****
15383: procedure SIRegister_cyIndy(CL: TPSPPascalCompiler);
15384: begin
15385:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15386:   +'__Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15387:   +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15388:   MessagePlainText', 'String 'text/plain');
15389:   CL.AddConstantN('MessagePlainText_Attach', 'String 'multipart/mixed');
15390:   MessageAlterText_Html', 'string 'multipart/alternative');
15391:   MessageHtml_Attach', 'String 'multipart/mixed');
15392:   MessageHtml_RelatedAttach', 'String 'multipart/related; type="text/html"');
15393:   MessageAlterText_Html_Attach', 'String 'multipart/mixed');
15394:   MessageAlterText_Html_RelatedAttach', 'String ')('multipart/related;type="multipart/alternative"');
15395:   MessageAlterText_Html_Attach_RelatedAttach', 'String 'multipart/mixed');
15396:   MessageReadNotification', 'String ')(('multipart/report; report-type="disposition-notification"';
15397:   Function ForceDecodeHeader( aHeader : String ) : String');
15398:   Function Base64_EncodeString( Value : String; const aEncoding : TEncoding ) : string');
15399:   Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding ) : String');
15400:   Function Base64_DecodeToBytes( Value : String ) : TBytes;');
15401:   Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15402:   Function Get_MD5( const aFileName : string ) : string');
15403:   Function Get_MD5FromString( const aString : string ) : string');
15404: end;
15405:
15406: procedure SIRegister_cySysUtils(CL: TPSPPascalCompiler);
15407: begin
15408:   Function IsFolder( SRec : TSearchrec ) : Boolean');
15409:   Function isFolderReadOnly( Directory : String ) : Boolean');
15410:   Function DirectoryIsEmpty( Directory : String ) : Boolean');
15411:   Function DirectoryWithSubDir( Directory : String ) : Boolean');
15412:   Procedure GetSubDirs( FromDirectory : String; aList : TStrings )');
15413:   Function DiskFreeBytes( Drv : Char ) : Int64');
15414:   Function DiskBytes( Drv : Char ) : Int64');
15415:   Function GetFileBytes( Filename : String ) : Int64');
15416:   Function GetFilesBytes( Directory, Filter : String ) : Int64');
15417:   SE_CREATE_TOKEN_NAME', 'String 'SeCreateTokenPrivilege');
15418:   SE_ASSIGNPRIMARYTOKEN_NAME', 'String 'SeAssignPrimaryTokenPrivilege');
15419:   SE_LOCK_MEMORY_NAME', 'String 'SeLockMemoryPrivilege');
15420:   SE_INCREASE_QUOTA_NAME', 'String 'SeIncreaseQuotaPrivilege');
15421:   SE_UNSOLICITED_INPUT_NAME', 'String 'SeUnsolicitedInputPrivilege');
15422:   SE_MACHINE_ACCOUNT_NAME', 'String 'SeMachineAccountPrivilege');
15423:   SE_TCB_NAME', 'String 'SeTcbPrivilege');

```

```

15424: SE_SECURITY_NAME', 'String 'SeSecurityPrivilege');
15425: SE_TAKE_OWNERSHIP_NAME', 'String 'SeTakeOwnershipPrivilege');
15426: SE_LOAD_DRIVER_NAME', 'String 'SeLoadDriverPrivilege');
15427: SE_SYSTEM_PROFILE_NAME', 'String 'SeSystemProfilePrivilege');
15428: SE_SYSTEMTIME_NAME', 'String 'SeSystemtimePrivilege');
15429: SE_PROF_SINGLE_PROCESS_NAME', 'String 'SeProfileSingleProcessPrivilege');
15430: SE_INC_BASE_PRIORITY_NAME', 'String 'SeIncreaseBasePriorityPrivilege');
15431: SE_CREATE_PAGEFILE_NAME', 'String 'SeCreatePagefilePrivilege');
15432: SE_CREATE_PERMANENT_NAME', 'String 'SeCreatePermanentPrivilege');
15433: SE_BACKUP_NAME', 'String 'SeBackupPrivilege');
15434: SE_RESTORE_NAME', 'String 'SeRestorePrivilege');
15435: SE_SHUTDOWN_NAME', 'String 'SeShutdownPrivilege');
15436: SE_DEBUG_NAME', 'String 'SeDebugPrivilege');
15437: SE_AUDIT_NAME', 'String 'SeAuditPrivilege');
15438: SE_SYSTEM_ENVIRONMENT_NAME', 'String 'SeSystemEnvironmentPrivilege');
15439: SE_CHANGE_NOTIFY_NAME', 'String 'SeChangeNotifyPrivilege');
15440: SE_REMOTE_SHUTDOWN_NAME', 'String 'SeRemoteShutdownPrivilege');
15441: SE_UNDOCK_NAME', 'String 'SeUndockPrivilege');
15442: SE_SYNC_AGENT_NAME', 'String 'SeSyncAgentPrivilege');
15443: SE_ENABLE_DELEGATION_NAME', 'String 'SeEnableDelegationPrivilege');
15444: SE_MANAGE_VOLUME_NAME', 'String 'SeManageVolumePrivilege');
15445: end;
15446:
15447: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15448: begin
15449:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15450:     + 'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15451:   Function ShellGetExtensionName( FileName : String ) : String';
15452:   Function ShellGetIconIndex( FileName : String ) : Integer';
15453:   Function ShellGetIconHandle( FileName : String ) : HIcon';
15454:   Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string );
15455:   Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string );
15456:   Procedure ShellRenameDir( DirFrom, DirTo : string );
15457:   Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15458:   Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15459:   Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String );
15460:   Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string );
15461:   Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
      WaitCloseCompletion : boolean) : Boolean';
15462:   Procedure RestoreAndSetForegroundWindow( Hnd : Integer );
15463:   Function RemoveDuplicatedPathDelimiter( Str : String ) : String';
15464:   Function cyFileTimeToDate( _FT : TFileTime ) : TDateTime';
15465:   Function GetModificationDate( Filename : String ) : TDateTime';
15466:   Function GetCreationDate( Filename : String ) : TDateTime';
15467:   Function GetLastAccessDate( Filename : String ) : TDateTime';
15468:   Function FileDelete( Filename : String ) : Boolean';
15469:   Function FileIsOpen( Filename : string ) : boolean';
15470:   Procedure FileDelete( FromDirectory : String; Filter : ShortString );
15471:   Function DirectoryDelete( Directory : String ) : Boolean';
15472:   Function GetPrinters( PrintersList : TStrings ) : Integer';
15473:   Procedure SetDefaultPrinter( PrinterName : String );
15474:   Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer );
15475:   Function WinToDosPath( WinPathName : String ) : String';
15476:   Function DosToWinPath( DosPathName : String ) : String';
15477:   Function cyGetWindowsVersion : TWindowsVersion';
15478:   Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean ) : Boolean';
15479:   Procedure WindowsShutDown( Restart : boolean );
15480:   Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
      FileIcon:string;NumIcone:integer);
15481:   Procedure GetWindowsFonts( FontsList : TStrings );
15482:   Function GetAvailableFilename( DesiredFileName : String ) : String';
15483: end;
15484:
15485: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15486: begin
15487:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15488:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15489:   Type(TStringRead', '( srFromLeft, srFromRight )');
15490:   Type(TStringReads', 'set of TStringRead');
15491:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15492:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15493:   Type(TWordsOptions', 'set of TWordsOption');
15494:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15495:   Type(TCarTypes', 'set of TCarType');
15496:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15497:   CarTypeAlphabetic'('LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15498:   Function Char_GetType( aChar : Char ) : TCarType';
15499:   Function SubString_Count( Str : String; Separator : Char ) : Integer';
15500:   Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15501:   Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word ) : String';
15502:   Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15503:   Procedure SubString_Add( var Str : String; Separator : Char; Value : String );
15504:   Procedure SubString_Insert(var Str:String;Separator:Char;SubStringIndex:Word;Value : String );
15505:   Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word;NewValue : String );
15506:   Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean';
15507:   Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15508:   Function SubString_Ribbon( Str : string; Separator : Char; Current : Word; MoveBy : Integer ): Integer';
15509:   Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String';
15510:   Function String_Quote( Str : String ) : String';

```

```

15511: Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char ) : Char';
15512: Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15513: Function String_GetWord( Str : String; StringRead : TStringRead ) : String');
15514: Function String_GetInteger( Str : String; StringRead : TStringRead ) : String');
15515: Function String_ToInt( Str : String ) : Integer');
15516: Function String_Uppercase( Str : String; Options : TWordsOptions ) : String');
15517: Function String_Lowercase( Str : String; Options : TWordsOptions ) : String');
15518: Function String_Reverse( Str : String ) : String');
15519: Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15520: Function String_Posl(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive : TCaseSensitive):Integer');
15521: Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer ) : String');
15522: Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String; Inclusive:Boolean):String;
15523: Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
Between2MustExist : Boolean; CaseSensitive : TCaseSensitive ) : String');
15524: Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer ) : String');
15525: Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String');
15526: Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads ) : String');
15527: Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer ) : String');
15528: Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String';
15529: Function String_End( Str : String; Cars : Word ) : String');
15530: Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
AlwaysFindFromBeginning : Boolean) : String');
15531: Function String_SubstCar( Str : String; Old, New : Char ) : String');
15532: Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive ) : Integer');
15533: Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15534: Function String_IsNumbers( Str : String ) : Boolean');
15535: Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer ) : Integer');
15536: Function StringToCsvCell( aStr : String ) : String');
15537: end;
15538:
15539: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15540: begin
15541:   Function LongDayName( aDate : TDate ) : String');
15542:   Function LongMonthName( aDate : TDate ) : String');
15543:   Function ShortYearOf( aDate : TDate ) : byte');
15544:   Function DateToStrYYYYMMDD( aDate : TDate ) : String');
15545:   Function StrYYYYMMDDToDate( aStr : String ) : TDate');
15546:   Function SecondsToMinutes( Seconds : Integer ) : Double');
15547:   Function MinutesToSeconds( Minutes : Double ) : Integer');
15548:   Function MinutesToHours( Minutes : Integer ) : Double');
15549:   Function HoursToMinutes( Hours : Double ) : Integer');
15550:   Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String ) : TDateTime');
15551:   Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer );
15552:   Function MergeDateWithTime( aDate : TDate; aTime : TDateTime ) : TDateTime');
15553:   Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime ) : Int64 );
15554:   Function GetMinutesBetween(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15555:   Function GetSecondsBetween( DateTime1, Datetime2 : TDateTime ) : Int64 );
15556:   Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
RsltBegin:TDateTime; RsltEnd : TDateTime ) : Boolean );
15557:   Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15558:   Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime ) : Boolean );
15559: end;
15560:
15561: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15562: begin
15563:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended ) ');
15564:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended ) ');
15565:   Type(TcyLocateOption', '( lCaseInsensitive, 1PartialKey ) ');
15566:   Type(TcyLocateOptions', 'set of TcyLocateOption' );
15567:   Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions ) : Integer');
15568:   Function StringsLocatel( aList : TStrings; Value : String; ValueKind : TStringsValueKind ) : Integer');
15569:   Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15570:   Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind );
15571:   Procedure StringsSort( aList : TStrings; SortType : TStringsSortType );
15572:   Function TreeNodeLocate( ParentNode : TTTreeNode; Value : String ) : TTTreeNode');
15573:   Function TreeNodeLocateOnLevel( TreeView : TTTreeView; OnLevel : Integer; Value : String ) : TTTreeNode');
15574:   Function
TreeNodeGetChildFromIndex(TreeView:TTTreeView;ParentNode:TTTreeNode;ChildIndex:Integer):TTTreeNode';
15575:   Function TreeNodeGetParentOnLevel( ChildNode : TTTreeNode; ParentLevel : Integer ) : TTTreeNode');
15576:   Procedure TreeNodeCopy(FromNode:TTTreeNode;ToNode:TTTreeNode;const CopyChildren:Boolean;const
CopySubChildren:Bool;
15577:   Procedure RichEditSetStr( aRichEdit : TRichEdit; FormattedString : String );
15578:   Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15579:   Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint ) : TControl );
15580:   Procedure cyCenterControl( aControl : TControl );
15581:   Function GetLastParent( aControl : TControl ) : TWinControl );
15582:   Function GetControlBitmap( aControl : TWinControl ) : TBitmap );
15583:   Function GetRichEditBitmap( aRichEdit : TRichEdit ) : TBitmap );
15584: end;
15585:
15586: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15587: begin
15588:   Function TablePackTable( Tab : TTable ) : Boolean );
15589:   Function TableRegenIndexes( Tab : TTable ) : Boolean );
15590:   Function TableShowDeletedRecords( Tab : TTable; Show : Boolean ) : Boolean );
15591:   Function TableUndeleteRecord( Tab : TTable ) : Boolean );
15592:   Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15593:   Function TableDeleteIndex( Tab : TTable; IndexFieldName : String ) : Boolean );

```

```

15594: Function TableEmptyTable( Tab : TTable ) : Boolean';
15595: Function TableFindKey( aTable : TTable; Value : String ) : Boolean';
15596: Procedure TableFindNearest( aTable : TTable; Value : String );
15597: Function
  TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Bool):TTable;
15598: Function
  TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15599: Function DateToBDESQLODate( aDate : TDate; const DateFormat : String ) : String';
15600: end;
15601;
15602: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15603: begin
15604:   SIRegister_TcyRunTimeDesign(CL);
15605:   SIRegister_TcyShadowText(CL);
15606:   SIRegister_TcyBgPicture(CL);
15607:   SIRegister_TcyGradient(CL);
15608:   SIRegister_TcyBevel(CL);
15609:   //CL.AddTypes('TcyBevelClass', 'class of tcyBevel');
15610:   SIRegister_tcyBevels(CL);
15611:   SIRegister_TcyImagelistOptions(CL);
15612:   Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture );
15613: end;
15614;
15615: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15616: begin
15617:   Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
    adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
    Maxdegrade : Byte );
15618:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap );
15619:   Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15620:   Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15621:   Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
    Byte; toRect : TRect; OrientationShape : TDgradOrientationShape );
15622:   Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
    Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap );
15623:   Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer );
15624:   Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer );
15625:   Procedure cyFrameL( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
    BottomColor : TColor; const Width : Integer; const RoundRect : boolean );
15626:   Procedure cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,
    BottomRightColor:TColor;Width:Integer;const DrawLeft:Boolean;const DrawTop:Boolean;const DrawRight:Boolean;const
    DrawBottom:Boolean;const RoundRect:bool);
15627:   Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
    aState : TButtonState; Focused, Hot : Boolean );
15628:   Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
    aState : TButtonState; Focused, Hot : Boolean );
15629:   Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
    aState : TButtonState; Focused, Hot : Boolean );
15630:   Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
    GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean );
15631:   Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
    const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor );
15632:   Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
    TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer );
15633:   Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
    TALignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15634:   Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
    WordWrap : Boolean; CaptionRender : TCaptionRender ) : LongInt );
15635:   Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt );
15636:   Function cyCreateFontIndirect( fromFont : TFont; Angle : Double ) : TFont;
15637:   Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation ) : TFont;
15638:   Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
    CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout );
15639:   Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone : Integer;
    const OnlyCalcFoldLine : Boolean ) : TLineCoord;
15640:   Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone : Integer;
    const OnlyCalcFoldLine : Boolean ) : TLineCoord;
15641:   Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint ) : boolean;
15642:   Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint ) : boolean;
15643:   Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint ) : boolean;
15644:   Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint ) : boolean;
15645:   Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect );
15646:   Procedure DrawCanvas1(Destination:TCanvas;
    DestRect:TRect;Src:TCanvas;SrcRect:TRect;TransparentColor:TColor; const aStyle:TBgStyle; const
    aPosition:TBgPosition; const IndentX : Integer; const IndentY : Integer;const IntervalX : Integer; const
    IntervalY : Integer; const RepeatX:Integer;const RepeatY:Integer);
15647:   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 );
15648:   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 );
15649:   Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap );
15650:   Function ValidGraphic( aGraphic : TGraphic ) : Boolean;
15651:   Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer ) : TColor;
15652:   Function ColorModify( Color : TColor; incR, incG, incB : Integer ) : TColor;
15653:   Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer ) : TColor;
15654:   Function ColorSetPercentPale( Color : TColor; IncPercent : integer ) : TColor;

```

```

15655: Function MediumColor( Color1, Color2 : TColor ) : TColor');
15656: Function ClientToScreenRect( aControl : TControl; aControlRect : TRect ) : TRect');
15657: Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect ) : TRect');
15658: Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect ) : TRect');
15659: Procedure InflateRectPercent( var aRect : TRect; withPercent : Double );
15660: Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double ) : TRect');
15661: Function GetProportionalRect( fromRect, InsideRect : TRect ) : TRect');
15662: Function PointInRect( const aPt : TPoint; const aRect : TRect ) : boolean );
15663: Function PointInEllispe( const aPt : TPoint; const aRect : TRect ) : boolean );
15664: Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String ) : Integer );
15665: end;
15666:
15667: procedure SIRegister_cyTypes(CL: TPPSPascalCompiler);
15668: begin
15669:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15670:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15671:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15672:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15673:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15674:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,
15675:     +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15676:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15677:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15678:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15679:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15680:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
15681:     bmInvertReverseFromColor));
15681:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
15682:     rjResizeTopLeft, rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15683:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15684:   cCaptionOrientationWarning', 'String')('Note that text orientation doesn''t work with all fonts! ');
15685:
15686: procedure SIRegister_WinSvc(CL: TPPSPascalCompiler);
15687: begin
15688:   Const SERVICES_ACTIVE_DATABASEA', 'String 'ServicesActive');
15689:   SERVICES_ACTIVE_DATABASEW', 'String') 'ServicesActive');
15690:   Const SERVICES_ACTIVE_DATABASE', 'String') SERVICES_ACTIVE_DATABASEA');
15691:   Const SERVICES_FAILED_DATABASEA', 'String') 'ServicesFailed');
15692:   Const SERVICES_FAILED_DATABASEW', 'String') 'ServicesFailed');
15693:   Const SERVICES_FAILED_DATABASE', 'String') SERVICES_FAILED_DATABASEA');
15694:   Const SC_GROUP_IDENTIFIERA', 'String'). '+');
15695:   Const SC_GROUP_IDENTIFIERW', 'String') '+');
15696:   Const SC_GROUP_IDENTIFIER', 'string SC_GROUP_IDENTIFIERA');
15697:   Const SERVICE_NO_CHANGE', 'LongWord $FFFFFF');
15698:   Const SERVICE_ACTIVE', 'LongWord')($00000001);
15699:   Const SERVICE_INACTIVE', 'LongWord $00000002);
15700:   Const SERVICE_CONTROL_STOP', 'LongWord $00000001);
15701:   Const SERVICE_CONTROL_PAUSE', 'LongWord $00000002);
15702:   Const SERVICE_CONTROL_CONTINUE', 'LongWord $00000003);
15703:   Const SERVICE_CONTROL_INTERROGATE', 'LongWord $00000004);
15704:   Const SERVICE_CONTROL_SHUTDOWN', 'LongWord $00000005);
15705:   Const SERVICE_STOPPED', 'LongWord $00000001);
15706:   Const SERVICE_START_PENDING', 'LongWord $00000002);
15707:   Const SERVICE_STOP_PENDING', 'LongWord $00000003);
15708:   Const SERVICE_RUNNING', 'LongWord $00000004);
15709:   Const SERVICE_CONTINUE_PENDING', 'LongWord $00000005);
15710:   Const SERVICE_PAUSE_PENDING', 'LongWord $00000006);
15711:   Const SERVICE_PAUSED', 'LongWord $00000007);
15712:   Const SERVICE_ACCEPT_STOP', 'LongWord $00000001);
15713:   Const SERVICE_ACCEPT_PAUSE_CONTINUE', 'LongWord $00000002);
15714:   Const SERVICE_ACCEPT_SHUTDOWN', 'LongWord $00000004);
15715:   Const SC_MANAGER_CONNECT', 'LongWord $0001);
15716:   Const SC_MANAGER_CREATE_SERVICE', 'LongWord $0002;
15717:   Const SC_MANAGER_ENUMERATE_SERVICE', 'LongWord $0004);
15718:   Const SC_MANAGER_LOCK', 'LongWord $0008);
15719:   Const SC_MANAGER_QUERY_LOCK_STATUS', 'LongWord $0010);
15720:   Const SC_MANAGER_MODIFY_BOOT_CONFIG', 'LongWord $0020);
15721:   Const SERVICE_QUERY_CONFIG', 'LongWord $0001);
15722:   Const SERVICE_CHANGE_CONFIG', 'LongWord $0002);
15723:   Const SERVICE_QUERY_STATUS', 'LongWord $0004);
15724:   Const SERVICE_ENUMERATE_DEPENDENTS', 'LongWord $0008);
15725:   Const SERVICE_START', 'LongWord $0010);
15726:   Const SERVICE_STOP', 'LongWord $0020);
15727:   Const SERVICE_PAUSE_CONTINUE', 'LongWord $0040);
15728:   Const SERVICE_INTERROGATE', 'LongWord $0080);
15729:   Const SERVICE_USER_DEFINED_CONTROL', 'LongWord $0100);
15730:   Const SERVICE_KERNEL_DRIVER', 'LongWord $00000001);
15731:   Const SERVICE_FILE_SYSTEM_DRIVER', 'LongWord $00000002);
15732:   Const SERVICE_ADAPTER', 'LongWord $00000004);
15733:   Const SERVICE_RECOGNIZER_DRIVER', 'LongWord $00000008);
15734:   Const SERVICE_WIN32_OWN_PROCESS', 'LongWord $00000010);
15735:   Const SERVICE_WIN32_SHARE_PROCESS', 'LongWord $00000020);
15736:   Const SERVICE_INTERACTIVE_PROCESS', 'LongWord $00000100);
15737:   Const SERVICE_BOOT_START', 'LongWord $00000000);
15738:   Const SERVICE_SYSTEM_START', 'LongWord $00000001);
15739:   Const SERVICE_AUTO_START', 'LongWord $00000002);
15740:   Const SERVICE_DEMAND_START', 'LongWord $00000003);
15741:   Const SERVICE_DISABLED', 'LongWord $00000004);

```

```

15742: Const SERVICE_ERROR_IGNORE', 'LongWord $00000000);
15743: Const SERVICE_ERROR_NORMAL', 'LongWord $00000001);
15744: Const SERVICE_ERROR_SEVERE', 'LongWord $00000002);
15745: Const SERVICE_ERROR_CRITICAL', 'LongWord $00000003);
15746: CL.AddTypeS('SC_HANDLE', 'THandle');
15747: //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15748: CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15749: Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15750: +' : DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15751: +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15752: Const SERVICE_STATUS', '_SERVICE_STATUS');
15753: Const TServiceStatus', '_SERVICE_STATUS');
15754: CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15755: +'playName : PChar; ServiceStatus : TServiceStatus; end');
15756: ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15757: _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15758: TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15759: TEnumServiceStatus', 'TEnumServiceStatusA');
15760: SC_LOCK', '__Pointer');
15761: _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOwner: PChar; dwLockDuration:DWORD; end';
15762: _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15763: QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15764: QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15765: TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15766: //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15767: TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15768: _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15769: +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15770: +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15771: +'iceStartName : PChar; lpDisplayName : PChar; end');
15772: _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15773: QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15774: QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15775: TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15776: TQueryServiceConfig', 'TQueryServiceConfigA');
15777: Function CloseServiceHandle( hSCObject : SC_HANDLE ) : BOOL';
15778: Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15779: Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;
15780: +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar ) : SC_HANDLE );
15781: Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;
15782: +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar ) : SC_HANDLE );
15783: Function DeleteService( hService : SC_HANDLE ) : BOOL';
15784: Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD ) : BOOL';
15785: Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState:DWORD;var lpServices:
TEnumServiceStatus;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD ):BOOL;
15786: Function GetServiceKeyName(hSCManager:SC_HANDLE;pdisplayName,lpServiceName:PChar;var
lpcchBuffer:DWORD):BOOL';
15787: Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
lpcchBuffer:DWORD):BOOL;
15788: Function LockServiceDatabase( hSCManager : SC_HANDLE ) : SC_LOCK );
15789: Function NotifyBootConfigStatus( BootAcceptable : BOOL ) : BOOL';
15790: Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD ) : SC_HANDLE );
15791: Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE );
15792: Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : BOOL';
15793: Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15794: Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15795: Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15796: Function UnlockServiceDatabase( ScLock : SC_LOCK ) : BOOL';
15797: end;
15798:
15799: procedure SIRegister_JvPickDate(CL: TPPSPascalCompiler);
15800: begin
15801: Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
AStrtOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek; AWeekendColor: TColor;
BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15802: Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
DlgCaption:TCaption;AStrtOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDateTime;
15803: Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15804: Function CreatePopupCalendar(AOwner: TComponent; ABiDiMode: TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):TWinControl;
15805: Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStrtOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15806: Function CreateNotifyThread(const
FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15807: end;
15808:
15809: procedure SIRegister_JclNTFS2(CL: TPPSPascalCompiler);
15810: begin
15811: CL.AddClassN(CL.FindClass('TOBJECT'), 'EJclNtfsError');
15812: CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15813: Function NtfsGetCompression2( const FileName : TFileName; var State : Short ) : Boolean;');
15814: Function NtfsGetCompression12( const FileName : TFileName ) : TFileCompressionState;');
15815: Function NtfsSetCompression2( const FileName : TFileName; const State : Short ) : Boolean;');
15816: Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15817: Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');

```

```

15818: Procedure NtfsSetDefaultFileCompression2( const Directory : string; const State:TFileCompressionState );
15819: Procedure NtfsSetPathCompression2( const Path:string;const State:TFileCompressionState;Recursive:Bool );
15820: Function NtfsSetSparse2( const FileName : string ) : Boolean';
15821: Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64 ) : Boolean';
15822: Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64 ) : Boolean';
15823: Function NtfsSparseStreamsSupported2( const Volume : string ) : Boolean';
15824: Function NtfsGetSparse2( const FileName : string ) : Boolean';
15825: Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD ) : Boolean';
15826: Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword ) : Boolean';
15827: Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD ) : Boolean';
15828: Function NtfsReparsePointsSupported2( const Volume : string ) : Boolean';
15829: Function NtfsFileHasReparsePoint2( const Path : string ) : Boolean';
15830: Function NtfsIsFolderMountPoint2( const Path : string ) : Boolean';
15831: Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char ) : Boolean';
15832: Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString ) : Boolean';
15833: CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15834: Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15835: Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15836: Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15837: Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15838: Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped ):Boolean';
15839: Function NtfsCreateJunctionPoint2( const Source, Destination : string ) : Boolean';
15840: Function NtfsDeleteJunctionPoint2( const Source : string ) : Boolean';
15841: Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15842: CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15843: +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile )');
15844: CL.AddTypeS('TStreamIds', 'set of TStreamId');
15845: TInternalFindStreamData', 'record FileHandle:THandle; Context : TObject; StreamIds:TStreamIds; end');
15846: CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15847: +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15848: Function NtfsFindFirstStream2( const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15849: Function NtfsFindNextStream2( var Data : TFindStreamData ) : Boolean';
15850: Function NtfsFindStreamClose2( var Data : TFindStreamData ) : Boolean';
15851: Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String ) : Boolean';
15852: Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString ) : Boolean';
15853: Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString ) : Boolean';
15854: CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15855: Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean';
15856: Function NtfsDeleteHardLinks2(const Path:string;const FileIdxHigh,FIdxLow:Card;const List:TStrings):Bool;
15857: Function NtfsDeleteHardLinks2( const FileName : string ) : Boolean';
15858: FindClass('TOBJECT'), 'TJclFileSummaryError');
15859: TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15860: TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15861: TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID ) : Boolean');
15862: CL.AddClassN(CL.FindClass('TOBJECT'), 'TJclFileSummary');
15863: SIRegister_TJclFilePropertySet(CL);
15864: //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15865: SIRegister_TJclFileSummary(CL);
15866: SIRegister_TJclDocSummaryInformation(CL);
15867: SIRegister_TJclImageSummaryInformation(CL);
15868: SIRegister_TJclMediaFileSummaryInformation(CL);
15869: SIRegister_TJclMSISummaryInformation(CL);
15870: SIRegister_TJclShellSummaryInformation(CL);
15871: SIRegister_TJclStorageSummaryInformation(CL);
15872: SIRegister_TJclImageSummaryInformation(CL);
15873: SIRegister_TJclDisplacedSummaryInformation(CL);
15874: SIRegister_TJclBriefCaseSummaryInformation(CL);
15875: SIRegister_TJclMiscSummaryInformation(CL);
15876: SIRegister_TJclWebViewSummaryInformation(CL);
15877: SIRegister_TJclMusicSummaryInformation(CL);
15878: SIRegister_TJclDRMSummaryInformation(CL);
15879: SIRegister_TJclVideoSummaryInformation(CL);
15880: SIRegister_TJclAudioSummaryInformation(CL);
15881: SIRegister_TJclControlPanelSummaryInformation(CL);
15882: SIRegister_TJclVolumeSummaryInformation(CL);
15883: SIRegister_TJclShareSummaryInformation(CL);
15884: SIRegister_TJclLinkSummaryInformation(CL);
15885: SIRegister_TJclQuerySummaryInformation(CL);
15886: SIRegister_TJclImageInformation(CL);
15887: SIRegister_TJclJpegSummaryInformation(CL);
15888: end;
15889:
15890: procedure SIRegister_Jcl8087(CL: TPPascalCompiler);
15891: begin
15892: AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15893: T8087Rounding', '( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15894: T8087Infinity', '( icProjective, icAffine )');
15895: T8087Exception', '( emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision,
15896: CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15897: Function Get8087ControlWord : Word');
15898: Function Get8087Infinity : T8087Infinity');
15899: Function Get8087Precision : T8087Precision');
15900: Function Get8087Rounding : T8087Rounding');
15901: Function Get8087StatusWord( ClearExceptions : Boolean ) : Word');
15902: Function Set8087Infinity( const Infinity : T8087Infinity ) : T8087Infinity');
15903: Function Set8087Precision( const Precision : T8087Precision ) : T8087Precision');
15904: Function Set8087Rounding( const Rounding : T8087Rounding ) : T8087Rounding');
15905: Function Set8087ControlWord( const Control : Word ) : Word');
15906: Function ClearPending8087Exceptions : T8087Exceptions');

```

```

15907: Function GetPending8087Exceptions : T8087Exceptions');
15908: Function GetMasked8087Exceptions : T8087Exceptions');
15909: Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15910: Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15911: Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15912: end;
15913:
15914: procedure SIRegister_JvBoxProcs(CL: TPSPPascalCompiler);
15915: begin
15916: Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15917: Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15918: Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool';
15919: Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15920: Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15921: Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15922: Function BoxGetFirstSelection( List : TWinControl) : Integer');
15923: Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15924: end;
15925:
15926: procedure SIRegister_UrlMon(CL: TPSPPascalCompiler);
15927: begin
15928: //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15929: //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15930: CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15931: type ULONG', 'Cardinal');
15932:     LPCWSTR', 'PChar');
15933: CL.AddTypeS('LPWSTR', 'PChar');
15934: LPSTR', 'PChar');
15935: TBindVerb', 'ULONG');
15936: TBindInfoF', 'ULONG');
15937: TBindF', 'ULONG');
15938: TBSFC', 'ULONG');
15939: TBindStatus', 'ULONG');
15940: TCIPStatus', 'ULONG');
15941: TBindString', 'ULONG');
15942: TPiFlags', 'ULONG');
15943: TOIBdgFlags', 'ULONG');
15944: TParseAction', 'ULONG');
15945: TPSUAction', 'ULONG');
15946: TQueryOption', 'ULONG');
15947: TPUAF', 'ULONG');
15948: TSZMFlags', 'ULONG');
15949: TUrlZone', 'ULONG');
15950: TUrlTemplate', 'ULONG');
15951: TZAFlags', 'ULONG');
15952: TUrlZoneReg', 'ULONG');
15953: 'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15954: CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15955: const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15956: const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15957: const 'CF_NULL','LongInt').SetInt( 0);
15958: const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15959: const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15960: const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15961: const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-bitmap');
15962: const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15963: const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15964: const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15965: const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15966: const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15967: const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15968: const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15969: const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15970: const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15971: const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15972: const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15973: const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15974: const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15975: const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15976: const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15977: const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15978: const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15979: const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15980: const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15981: const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15982: const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15983: const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15984: const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15985: const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15986: const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15987: const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15988: const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15989: const 'MK_S_ASYNCNROUS','LongWord').SetUInt( $000401E8);
15990: const 'S_ASYNCNROUS','LongWord').SetUInt( $000401E8);
15991: const 'E_PENDING','LongWord').SetUInt( $8000000A);
15992: CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15993: SIRegister_IPersistMoniker(CL);
15994: SIRegister_IBindProtocol(CL);

```

```

15995:     SIRegister_IBinding(CL);
15996:     const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15997:     const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15998:     const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15999:     const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
16000:     const 'BINDINFO_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
16001:     const 'BINDINFO_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
16002:     const 'BINDF_ASYNCRONOUS','LongWord').SetUInt( $00000001);
16003:     const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
16004:     const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
16005:     const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
16006:     const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
16007:     const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
16008:     const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
16009:     const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
16010:     const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
16011:     const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
16012:     const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
16013:     const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
16014:     const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
16015:     const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
16016:     const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
16017:     const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
16018:     const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
16019:     const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
16020: //const 'BINDF_DONTUSECACHE','','').SetString( BINDF_GETNEWESTVERSION);
16021: //const 'BINDF_DONTPUTINCACHE','','').SetString( BINDF_NOWRITECACHE);
16022: //const 'BINDF_NOCOPYDATA','','').SetString( BINDF_PULLDATA);
16023: const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
16024: const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
16025: const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
16026: const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
16027: const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
16028: const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
16029: const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16030: const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16031: const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16032: const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16033: const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16034: const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
16035: const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
16036: const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
16037: const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16038: const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16039: const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16040: const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16041: const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
16042: const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
16043: const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
16044: const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16045: const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16046: const 'BINDSTATUS_ENDDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16047: const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDDUPLOADDATA + 1);
16048: const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16049: const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16050: const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
16051: const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16052: const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16053: const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
16054: const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
16055: const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
16056: const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16057: const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16058: const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16059: const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16060: const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16061: const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16062: const 'BINDSTATUS_COMPACT_POLICY RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16063: const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY RECEIVED + 1);
16064: const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16065: const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
16066: const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16067: const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
16068: const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16069: const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
16070: const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16071: const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16072: const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16073: const 'BINDSTATUS_PERSISTENT_COOKIE RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
16074: const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE RECEIVED + 1);
16075: const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16076: const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
16077: const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
16078: const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
16079: const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16080: // PBindInfo', '^TBindInfo // will not work');
16081: {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16082: +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16083: +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'

```

```

16084: +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16085: +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16086: TBindInfo', '_tagBINDINFO');
16087: BINDINFO', '_tagBINDINFO');}
16088: _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16089: +'criptor : DWORD; bInheritHandle : BOOL; end');
16090: TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
16091: REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
16092: //PREmBindInfo', '^TRemBindInfo // will not work');
16093: {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16094: +'grfBindInfo : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16095: +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16096: +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16097: +'n; dwReserved : DWORD; end');
16098: TRemBindInfo', '_tagRemBINDINFO');
16099: RemBINDINFO', '_tagRemBINDINFO');
16100: //PREmFormatEtc', '^TRemFormatEtc // will not work');
16101: tagRemFORMATETC', 'record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16102: TRemFormatEtc', 'tagRemFORMATETC');
16103: RemFORMATETC', 'tagRemFORMATETC');
16104: SIRegister_IBindStatusCallback(CL);
16105: SIRegister_IAuthenticate(CL);
16106: SIRegister_IHttpNegotiate(CL);
16107: SIRegister_IWindowForBindingUI(CL);
16108: const 'CIP_DISK_FULL','LongInt').SetInt( 0 );
16109: const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1 );
16110: const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1 );
16111: const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1 );
16112: const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1 );
16113: const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1 );
16114: const 'CIP_EXE_SELF_REGISTERATION_TIMEOUT','LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1 );
16115: const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1 );
16116: const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1 );
16117: const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1 );
16118: SIRegister_ICodeInstall(CL);
16119: SIRegister_IWInetInfo(CL);
16120: const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534 );
16121: SIRegister_IHttpSecurity(CL);
16122: SIRegister_IWInetHttpInfo(CL);
16123: SIRegister_IBindHost(CL);
16124: const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001 );
16125: const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002 );
16126: const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003 );
16127: Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback : HResult );
16128: Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback : HResult );
16129: Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
: IBindStatusCallback ) : HResult );
16130: Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
IBindStatusCallback ) : HResult );
16131: Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult';
16132: Function HlinkGoBack( unk : IUnknown ) : HResult';
16133: Function HlinkGoForward( unk : IUnknown ) : HResult';
16134: Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR ) : HResult';
16135: // Function HlinkNavigateMoniker( unk : IUnknown; mkTarget : IMoniker ) : HResult );
16136: SIRegister_IInternet(CL);
16137: const 'BINDSTRING_HEADERS','LongInt').SetInt( 1 );
16138: const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1 );
16139: const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1 );
16140: const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1 );
16141: const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1 );
16142: const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1 );
16143: const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1 );
16144: const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1 );
16145: const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1 );
16146: const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1 );
16147: const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1 );
16148: const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1 );
16149: const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1 );
16150: const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1 );
16151: //POLEStrArray', '^TOLESTRArray // will not work';
16152: SIRegister_IInternetBindInfo(CL);
16153: const 'PI_PARSE_URL','LongWord').SetUInt( $00000001 );
16154: const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002 );
16155: const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004 );
16156: const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008 );
16157: const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010 );
16158: const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020 );
16159: const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040 );
16160: const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080 );
16161: const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100 );
16162: const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200 );
16163: const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000 );
16164: //const 'PI_DOCFILECLSIDLOOKUP','','').SetString( PI_CLSIDLOOKUP );
16165: //PProtocolData', '^TProtocolData // will not work');
16166: _tagPROTOCOLDATA', 'record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16167: TProtocolData', '_tagPROTOCOLDATA');
16168: PROTOCOLDATA', '_tagPROTOCOLDATA');
16169: CL.AddInterface(CL.FindInterface('UNKNOWN'),IIInternetProtocolSink, 'IIInternetProtocolSink');


```

```

16170: SIRegister_IInternetProtocolRoot(CL);
16171: SIRegister_IInternetProtocol(CL);
16172: SIRegister_IInternetProtocolSink(CL);
16173: const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16174: SIRegister_IInternetSession(CL);
16175: SIRegister_IInternetThreadSwitch(CL);
16176: SIRegister_IInternetPriority(CL);
16177: const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16178: const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16179: const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16180: const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16181: const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16182: const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16183: const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16184: const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16185: const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16186: const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16187: const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16188: const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16189: const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16190: const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16191: const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16192: const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16193: const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16194: const 'PSU_DEFAULT','LongInt').SetInt( 1);
16195: const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16196: const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16197: const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16198: const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16199: const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16200: const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16201: const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16202: const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16203: const 'QUERYUSES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16204: const 'QUERY_IS_CACHED','LongInt').SetInt( QUERYUSES_NETWORK + 1);
16205: const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16206: const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16207: const 'QUERYUSES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16208: const 'QUERY_IS_SECURE','LongInt').SetInt( QUERYUSES_CACHE + 1);
16209: const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16210: SIRegister_IInternetProtocolInfo(CL);
16211: IOInet', 'IInternet');
16212: IOInetBindInfo', 'IInternetBindInfo');
16213: IOInetProtocolRoot', 'IInternetProtocolRoot');
16214: IOInetProtocol', 'IInternetProtocol');
16215: IOInetProtocolSink', 'IInternetProtocolSink');
16216: IOInetProtocolInfo', 'IInternetProtocolInfo');
16217: IOInetSession', 'IInternetSession');
16218: IOInetPriority', 'IInternetPriority');
16219: IOInetThreadSwitch', 'IInternetThreadSwitch');
16220: Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16221: Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16222: Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult';
16223: Function CoInternetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags DWORD; dwReserved:DWORD):HResult';
16224: Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult';
16225: Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession: IInternetSes;dwReserved:DWORD):HResult;
16226: Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16227: Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16228: Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16229: Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult';
16230: Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult';
16231: Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult';
16232: Function OInetGetSession(dwSessionMode:DWORD; var pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16233: //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult';
16234: //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16235: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult( $800C0011 ) );
16236: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult( $800C0012 ) );
16237: //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult( $800C0011 ) );
16238: //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult( $800C0013 ) );
16239: //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult( $800C0014 ) );
16240: const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16241: SIRegister_IInternetSecurityMgrSite(CL);
16242: const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16243: const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16244: const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16245: const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16246: const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16247: const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16248: const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16249: const 'PUAF_NOUI','LongWord').SetUInt( $00000001);

```

```

16250: const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16251: const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16252: const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16253: const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16254: const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16255: const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16256: const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16257: const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16258: const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16259: const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16260: const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16261: const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16262: const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16263: const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16264: const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16265: const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16266: const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16267: const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16268: const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16269: SIRegister_IInternetSecurityManager(CL);
16270: SIRegister_IInternetHostSecurityManager(CL);
16271: SIRegister_IInternetSecurityManagerEx(CL);
16272: const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16273: const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16274: const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16275: const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16276: const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16277: const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16278: const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16279: const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16280: const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16281: const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16282: const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16283: const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16284: const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16285: const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16286: const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16287: const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16288: const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16289: const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16290: const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16291: const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16292: const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16293: const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16294: const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16295: const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16296: const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16297: const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16298: const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16299: const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16300: const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16301: const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16302: const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16303: const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16304: const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16305: const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16306: const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16307: const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16308: const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16309: const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16310: const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16311: const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16312: const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16313: const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16314: const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16315: const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16316: const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16317: const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16318: const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16319: const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16320: const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16321: const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16322: const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16323: const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16324: const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16325: const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16326: const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16327: const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16328: const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16329: const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16330: const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16331: const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16332: const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16333: const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16334: const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16335: const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16336: const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16337: const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16338: const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);

```

```

16339: const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS', 'LongWord').SetUInt( $00001D01);
16340: const 'URLACTION_INFODELIVERY_NO_Removing_CHANNELS', 'LongWord').SetUInt( $00001D02);
16341: const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS', 'LongWord').SetUInt( $00001D03);
16342: const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS', 'LongWord').SetUInt( $00001D04);
16343: const 'URLACTION_INFODELIVERY_NO_Removing_SUBSCRIPTIONS', 'LongWord').SetUInt( $00001D05);
16344: const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING', 'LongWord').SetUInt( $00001D06);
16345: const 'URLACTION_INFODELIVERY_CURR_MAX', 'LongWord').SetUInt( $00001D06);
16346: const 'URLACTION_INFODELIVERY_MAX', 'LongWord').SetUInt( $00001Dff);
16347: const 'URLACTION_CHANNEL_SOFTDIST_MIN', 'LongWord').SetUInt( $00001E00);
16348: const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS', 'LongWord').SetUInt( $00001E05);
16349: const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT', 'LongWord').SetUInt( $00010000);
16350: const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE', 'LongWord').SetUInt( $00020000);
16351: const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL', 'LongWord').SetUInt( $00030000);
16352: const 'URLACTION_CHANNEL_SOFTDIST_MAX', 'LongWord').SetUInt( $00001EFF);
16353: const 'URLACTION_BEHAVIOR_MIN', 'LongWord').SetUInt( $00002000);
16354: const 'URLACTION_BEHAVIOR_RUN', 'LongWord').SetUInt( $00002000);
16355: const 'URLPOLICY_BEHAVIOR_CHECK_LIST', 'LongWord').SetUInt( $00001000);
16356: const 'URLACTION_FEATURE_MIN', 'LongWord').SetUInt( $00002100);
16357: const 'URLACTION_FEATURE_MIME_SNIFFING', 'LongWord').SetUInt( $00002100);
16358: const 'URLACTION_FEATURE_ZONE_ELEVATION', 'LongWord').SetUInt( $00002101);
16359: const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS', 'LongWord').SetUInt( $00002102);
16360: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN', 'LongWord').SetUInt( $00002200);
16361: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI', 'LongWord').SetUInt( $00002200);
16362: const 'URLACTION_AUTOMATIC_ACTIVEX_UI', 'LongWord').SetUInt( $00002201);
16363: const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS', 'LongWord').SetUInt( $00002300);
16364: const 'URLPOLICY_ALLOW', 'LongWord').SetUInt( $00);
16365: const 'URLPOLICY_QUERY', 'LongWord').SetUInt( $01);
16366: const 'URLPOLICY_DISALLOW', 'LongWord').SetUInt( $03);
16367: const 'URLPOLICY_NOTIFY_ON_ALLOW', 'LongWord').SetUInt( $10);
16368: const 'URLPOLICY_NOTIFY_ON_DISALLOW', 'LongWord').SetUInt( $20);
16369: const 'URLPOLICY_LOG_ON_ALLOW', 'LongWord').SetUInt( $40);
16370: const 'URLPOLICY_LOG_ON_DISALLOW', 'LongWord').SetUInt( $80);
16371: const 'URLPOLICY_MASK_PERMISSIONS', 'LongWord').SetUInt( $0f);
16372: Function GetUrlPolicyPermissions( dw : DWORD) : DWORD';
16373: Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD';
16374: const 'URLZONE_PREDEFINED_MIN', 'LongInt').SetInt( 0);
16375: const 'URLZONE_LOCAL_MACHINE', 'LongInt').SetInt( 0);
16376: const 'URLZONE_INTRANET', 'LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16377: const 'URLZONE_TRUSTED', 'LongInt').SetInt( URLZONE_INTRANET + 1);
16378: const 'URLZONE_INTERNET', 'LongInt').SetInt( URLZONE_TRUSTED + 1);
16379: const 'URLZONE_UNTRUSTED', 'LongInt').SetInt( URLZONE_INTERNET + 1);
16380: const 'URLZONE_PREDEFINED_MAX', 'LongInt').SetInt( 999);
16381: const 'URLZONE_USER_MIN', 'LongInt').SetInt( 1000);
16382: const 'URLZONE_USER_MAX', 'LongInt').SetInt( 10000);
16383: const 'URLTEMPLATE_CUSTOM', 'LongWord').SetUInt( $00000000);
16384: const 'URLTEMPLATE_PREAMBLE', 'LongWord').SetUInt( $00010000);
16385: const 'URLTEMPLATE_LOW', 'LongWord').SetUInt( $00010000);
16386: const 'URLTEMPLATE_MEDIUM', 'LongWord').SetUInt( $00011000);
16387: const 'URLTEMPLATE_HIGH', 'LongWord').SetUInt( $00012000);
16388: const 'URLTEMPLATE_PREAMBLE_MAX', 'LongWord').SetUInt( $00020000);
16389: const 'MAX_ZONE_PATH', 'LongInt').SetInt( 260);
16390: const 'MAX_ZONE_DESCRIPTION', 'LongInt').SetInt( 200);
16391: const 'ZAFLAGS_CUSTOM_EDIT', 'LongWord').SetUInt( $00000001);
16392: const 'ZAFLAGS_ADD_SITES', 'LongWord').SetUInt( $00000002);
16393: const 'ZAFLAGS_REQUIRE_VERIFICATION', 'LongWord').SetUInt( $00000004);
16394: const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE', 'LongWord').SetUInt( $00000008);
16395: const 'ZAFLAGS_INCLUDE_INTRANET_SITES', 'LongWord').SetUInt( $00000010);
16396: const 'ZAFLAGS_NO_UI', 'LongWord').SetUInt( $00000020);
16397: const 'ZAFLAGS_SUPPORTS_VERIFICATION', 'LongWord').SetUInt( $00000040);
16398: const 'ZAFLAGS_UNC_AS_INTRANET', 'LongWord').SetUInt( $00000080);
16399: const 'ZAFLAGS_USE_LOCKED_ZONES', 'LongWord').SetUInt( $00010000);
16400: //PZoneAttributes', '_TZoneAttributes // will not work');
16401: _ZONEATTRIBUTES', 'record cbSize:ULONG;szDisplayName:array [0..260-1] of Char;szDescription:array [0..200 - 1] of Char;szIconPath: array [0..260 - 1] of char; dwTemplateMinLevel: DWORD; dwTemplateRecommended: DWORD; dwTemplateCurrentLevel: DWORD; dwFlags: DWORD; end');
16402: { _ZONEATTRIBUTES = packed record
16403:   cbSize: ULONG;
16404:   szDisplayName: array [0..260 - 1] of WideChar;
16405:   szDescription: array [0..200 - 1] of WideChar;
16406:   szIconPath: array [0..260 - 1] of WideChar;
16407:   dwTemplateMinLevel: DWORD;
16408:   dwTemplateRecommended: DWORD;
16409:   dwTemplateCurrentLevel: DWORD;
16410:   dwFlags: DWORD;
16411: end; }
16412: TZoneAttributes', '_ZONEATTRIBUTES');
16413: ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16414: const 'URLZONEREG_DEFAULT', 'LongInt').SetInt( 0);
16415: const 'URLZONEREG_HKLM', 'LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16416: const 'URLZONEREG_HKCU', 'LongInt').SetInt( URLZONEREG_HKLM + 1);
16417: SIRegister_IInternetZoneManager(CL);
16418: SIRegister_IInternetZoneManagerEx(CL);
16419: const 'SOFTDIST_FLAG_USAGE_EMAIL', 'LongWord').SetUInt( $00000001);
16420: const 'SOFTDIST_FLAG_USAGE_PRECACHE', 'LongWord').SetUInt( $00000002);
16421: const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL', 'LongWord').SetUInt( $00000004);
16422: const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION', 'LongWord').SetUInt( $00000008);
16423: const 'SOFTDIST_ADSTATE_NONE', 'LongWord').SetUInt( $00000000);
16424: const 'SOFTDIST_ADSTATE_AVAILABLE', 'LongWord').SetUInt( $00000001);
16425: const 'SOFTDIST_ADSTATE_DOWNLOADED', 'LongWord').SetUInt( $00000002);

```

```

16426: const 'SOFTDIST_ADSTATE_INSTALLED', 'LongWord').SetUInt( $00000003);
16427: //PCodeBaseHold', '^TCodeBaseHold // will not work');
16428: _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR;
16429: + 'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16430: TCodeBaseHold', '_tagCODEBASEHOLD');
16431: CODEBASEHOLD', '_tagCODEBASEHOLD');
16432: //PSoftDistInfo', '^TSoftDistInfo // will not work');
16433: _tagsOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd';
16434: +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI';
16435: +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS';
16436: +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve';
16437: +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16438: TSoftDistInfo', '_tagSOFHTDISTINFO');
16439: SOFTDISTINFO', '_tagSOFHTDISTINFO');
16440: SIRegister_ISoftDistExt(CL);
16441: Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16442: Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
dwAdvertisedVersionLS : DWORD) : HResult');
16443: SIRegister_IDataFilter(CL);
16444: // PProtocolFilterData', '^TProtocolFilterData // will not work');
16445: _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolsSink : '
16446: +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil';
16447: +'terFlags : DWORD; end');
16448: TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16449: PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16450: //PDataInfo', '^TDataInfo // will not work');
16451: _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL';
16452: +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16453: TDataInfo', '_tagDATAINFO');
16454: DATAINFO', '_tagDATAINFO');
16455: SIRegister_IEncodingFilterFactory(CL);
16456: Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16457: //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16458: //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16459: //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16460: _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU';
16461: +'rlName : LPSTR; StartTime : TSystemTime; EndTime: TSystemTime; lpszExtendedInfo : LPSTR; end');
16462: THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16463: HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16464: Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16465: end;
16466:
16467: procedure SIRegister_DFFUtils(CL: TPPascalCompiler);
16468: begin
16469: Procedure reformatMemo( const m : TCustomMemo)');
16470: Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16471: Procedure MoveToTop( memo : TMemo)');
16472: Procedure ScrollToTop( memo : TMemo)');
16473: Function LineNumberClicked( memo : TMemo) : integer');
16474: Function MemoClickedLine( memo : TMemo) : integer');
16475: Function ClickedMemoLine( memo : TMemo) : integer');
16476: Function MemoLineClicked( memo : TMemo) : integer');
16477: Function LinePositionClicked( Memo : TMemo) : integer');
16478: Function ClickedMemoPosition( memo : TMemo) : integer');
16479: Function MemoPositionClicked( memo : TMemo) : integer');
16480: Procedure AdjustGridSize( grid : TDrawGrid)');
16481: Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16482: Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16483: Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer)');
16484: Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascendant : boolean;');
16485: Procedure sortstrDown( var s : string)');
16486: Procedure sortstrUp( var s : string)');
16487: Procedure rotatestrleft( var s : string)');
16488: Function dffstrtofloatdef( s : string; default : extended) : extended');
16489: Function deblank( s : string) : string');
16490: Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16491: Procedure FreeAndClearListBox( C : TListBox)');
16492: Procedure FreeAndClearMemo( C : TMemo)');
16493: Procedure FreeAndClearStringList( C : TStringList)');
16494: Function dffgetfilesize( f : TSearchrec) : int64');
16495: end;
16496:
16497: procedure SIRegister_MathLib(CL: TPPascalCompiler);
16498: begin
16499: CL.AddTypeS('intset', 'set of byte');
16500: TPoint64', 'record x : int64; y : int64; end');
16501: Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16502: Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16503: Function GeneratePentagon( n : integer) : integer');
16504: Function IsPentagon( p : integer) : boolean');
16505: Function isSquare( const N : int64) : boolean');
16506: Function isCube( const N : int64) : boolean');
16507: Function isPalindrome( const n : int64) : boolean');
16508: Function isPalindrome1( const n : int64; var len : integer) : boolean');
16509: Function GetEulerPhi( n : int64) : int64');
16510: Function dffIntPower( a, b : int64) : int64;');
16511: Function IntPower1( a : extended; b : int64) : extended;');
16512: Function gcd2( a, b : int64) : int64');
16513: Function GCDMany( A : array of integer) : integer');

```

```

16514: Function LCMMany( A : array of integer ) : integer');
16515: Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16516: Function dffFactorial( n : int64 ) : int64';
16517: Function digitcount( n : int64 ) : integer';
16518: Function nextpermute( var a : array of integer ) : boolean');
16519: Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16520: Function convertStringToDecimal( s : string; var n : extended ) : Boolean';
16521: Function InttoBinaryStr( nn : integer ) : string';
16522: Function StrtoAngle( const s : string; var angle : extended ) : boolean');
16523: Function AngleToStr( angle : extended ) : string');
16524: Function deg2rad( deg : extended ) : extended');
16525: Function rad2deg( rad : extended ) : extended');
16526: Function GetLongToMercProjection( const long : extended ) : extended');
16527: Function GetLatToMercProjection( const Lat : Extended ) : Extended');
16528: Function GetMercProjectionToLong( const ProjLong : extended ) : extended');
16529: Function GetMercProjectionToLat( const ProjLat : extended ) : extended');
16530: SIRegister_TPrimes(CL);
16531: //RIRegister_TPrimes(CL);
16532: //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16533: CL.AddConstantN('minmark','LongInt').SetInt( 180);
16534: Function Random64( const N : Int64 ) : Int64;');
16535: Procedure Randomize64');
16536: Function Random641 : extended;');
16537: Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUUtils
16538: end;
16539:
16540: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16541: begin
16542:   TrealPoint', 'record x : extended; y : extended; end');
16543:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16544:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16545:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16546:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16547:   PPResult', '( PPOutside, PPIInside, PPVertex, PPEdge, PPError )');
16548:   Function realpoint( x, y : extended ) : TRealPoint');
16549:   Function dist( const p1, p2 : TrealPoint ) : extended');
16550:   Function intdist( const p1, p2 : TPoint ) : integer');
16551:   Function dffLine( const p1, p2 : TPoint ) : Tline;');
16552:   Function Linel( const p1, p2 : TRealPoint ) : TRealline');
16553:   Function dffCircle( const cx, cy, R : integer ) : TCircle');
16554:   Function Circlel( const cx, cy, R : extended ) : TRealCircle');
16555:   Function GetTheta( const L : TLine ) : extended');
16556:   Function GetTheta1( const p1, p2 : TPoint ) : extended');
16557:   Function GetTheta2( const p1, p2 : TRealPoint ) : extended');
16558:   Procedure Extendline( var L : TLine; dist : integer );
16559:   Procedure Extendline1( var L : TRealLine; dist : extended );
16560:   Function Linesintersect( line1, line2 : TLine ) : boolean');
16561:   Function ExtendedLinesIntersect( Line1,Line2:TLine; const extendlines:bool;var IP:TPoint ):bool;
16562:   Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint ):bool;
16563:   Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint ) : boolean');
16564:   Function PointPerpendicularLine( L : TLine; P : TPoint ) : TLine');
16565:   Function PerpDistance( L : TLine; P : TPoint ) : Integer');
16566:   Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended ) : TLine');
16567:   Function AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16568:   Function PointInPoly( const p : TPoint; Points : array of TPoint ) : PPResult');
16569:   Function PolygonArea( const points:array of TPoint;const screencoordinates:boolean;var Clockwise:boolean ) : integer;
16570:   Procedure InflatePolygon( const points:array of Tpoint; var points2:array of TPoint;var area:integer;
16571:     const screenCoordinates : boolean; const inflateby : integer );
16572:   Function PolyBuiltClockwise( const points:array of TPoint;const screencoordinates:boolean ) :bool;
16573:   Function DegtoRad( d : extended ) : extended');
16574:   Function RadtoDeg( r : extended ) : extended');
16575:   Procedure TranslateLeftTo( var L : TLine; newend : TPoint );
16576:   Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint );
16577:   Procedure RotateRightEndBy( var L : TLine; alpha : extended );
16578:   Procedure RotateRightEndTo( var L : Tline; alpha : extended );
16579:   Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint ) : boolean );
16580:   Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint ) : boolean );
16581:   Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine ) : boolean );
16582:   Function CircleCircleExtTangentLines(C1,C2:TCircle;var L1,L2,PL1,PL2,TL1,TL2:TLine):Bool;
16583: end;
16584:
16585:
16586: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16587: begin
16588:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGallonLat )');
16589:   TDType', '( ttLocal, ttUT, ttGST, ttLST )');
16590:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16591:   TSunrec', 'record TrueEclLon:extended;
16592:     AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
16593:     TrueAzAlt:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRPoint;SunMeanAnomaly:extended;end;
16594:     TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl : '
16595:       +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16596:       +' arth : extended; Phase : extended; end');
16597:     TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16598:       +' Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16599:     TEclipseRec', 'record Msg : string; Status : integer; FirstConta'

```

```

16598:   +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime:end');
16599:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16600:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16601:     +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16602:     +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16603:     +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16604:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector : '
16605:     extended; UncorrectedEarthDistance : extended; GeoEclLonLat : TRpoint; CorrectedEarthDistance:extended;
16606:     ApparentRaDecl:TRPoint; end');
16607:   SIRegister_TAstronomy(CL);
16608:   Function AngleToStr( angle : extended) : string');
16609:   Function StrToAngle( s : string; var angle : extended) : boolean');
16610:   Function HoursToStr24( t : extended) : string');
16611:   Function RPoint( x, y : extended) : TRPoint');
16612:   Function getStimenename( t : TDType) : string');
16613: end;
16614: procedure SIRegister_UCardComponentV2(CL: TPSPPascalCompiler);
16615: begin
16616:   TCardValue', 'Integer');
16617:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16618:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16619:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16620:   SIRegister_TCard(CL);
16621:   SIRegister_TDeck(CL);
16622: end;
16623: procedure SIRegister_UTGraphSearch(CL: TPSPPascalCompiler);
16624: begin
16625:   tMethodCall', 'Procedure');
16626:   tVerboseCall', 'Procedure ( s : string)');
16627:   // PTEdge', '^TEdge // will not work');
16628:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16629:     +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16630:   SIRegister_TNode(CL);
16631:   SIRegister_TGraphList(CL);
16632: end;
16633: procedure SIRegister_UParser10(CL: TPSPPascalCompiler);
16634: begin
16635:   ParserFloat', 'extended');
16636:   //PParserFloat', '^ParserFloat // will not work');
16637:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16638:   +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16639:   //POperation', '^TOperation // will not work');
16640:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16641:     +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure';
16642:     +'; Token : TDFFToken; end');
16643:   TMathProcedure', 'procedure(AnOperation: TDFFOperation)');
16644:   (CL.FindClass('TOBJECT'),'EMathParserError');
16645:   CL.FindClass('TOBJECT','ESyntaxError');
16646:   CL.FindClass('TOBJECT','EExpressionHasBlanks');
16647:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16648:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16649:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16650:   (CL.FindClass('TOBJECT'),'EBadName');
16651:   (CL.FindClass('TOBJECT'),'EParseInternalError');
16652:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16653:   SIRegister_TCustomParser(CL);
16654:   SIRegister_TExParser(CL);
16655: end;
16656: function isService: boolean;
16657: begin
16658:   result:= NOT(Application is TApplication);
16659:   {result:= Application is TServiceApplication;}
16660: end;
16661: function isApplication: boolean;
16662: begin
16663:   result:= Application is TApplication;
16664: end;
16665: //SM_REMOTESESSION = $1000
16666: function isTerminalSession: boolean;
16667: begin
16668:   result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16669: end;
16670: procedure SIRegister_cyIEUtils(CL: TPSPPascalCompiler);
16671: begin
16672:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16673:     +'String; margin_bottom : String; margin_left : String; margin_right : String'
16674:     +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16675:   Function cyURLEncode( const S : string ) : string');
16676:   Function MakeResourceURL( const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16677:   Function MakeResourceURL1( const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;
16678:   Function cyColorToHtml( aColor : TColor ) : String';
16679:   Function HtmlToColor( aHtmlColor : String ) : TColor';
16680:   //Function GetStreamEncoding( aStream : TStream ) : TEncoding';
16681:   // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding ) : Boolean');
16682: 
```

```

16685: Function AddHtmlUnicodePrefix( aHtml : String ) : String';
16686: Function RemoveHtmlUnicodePrefix( aHtml : String ) : String';
16687: Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup );
16688: Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup );
16689: CL.AddConstantN('IEBodyBorderless','String').SetString('none');
16690: CL.AddConstantN('IEBodySingleBorder','String').SetString('');
16691: CL.AddConstantN('IEDesignModeOn','String').SetString('On');
16692: CL.AddConstantN('IEDesignModeOff','String').SetString('Off');
16693: CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF );
16694: CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE );
16695: end;
16696:
16697:
16698: procedure SIRегистер_UComboV2(CL: TPSPascalCompiler);
16699: begin
16700:   CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600 );
16701:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16702:     + 'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16703:     + 'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16704:     + 'inationsRepeat, CombinationsRepeatDown )');
16705:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16706:   SIRегистер_TComboSet(CL);
16707: end;
16708:
16709: procedure SIRегистер_cyBaseComm(CL: TPSPascalCompiler);
16710: begin
16711:   TcyCommandType', '( ctDeveloperDefined, ctUserDefined )');
16712:   TStreamContentType', '( scDeveloperDefined, scUserDefined, scString )');
16713:   TProcOnReceiveCommand', 'Procedure ( Sender: TObject; aCommand: Word; userParam : Integer )';
16714:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16715:     + 'mmHandle : THandle; aString : String; userParam : Integer )';
16716:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16717:     + 'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer )';
16718:   SIRегистер_TcyBaseComm(CL);
16719:   CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1 );
16720:   CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99 );
16721:   Function ValidatefileMappingName( aName : String ) : String';
16722:   procedure makeCaption(leftSide, Rightside:string; form:TForm);
16723:   end;
16724:
16725: procedure SIRегистер_cyDERUtils(CL: TPSPascalCompiler);
16726: begin
16727:   CL.AddTypeS('DERString', 'String');
16728:   CL.AddTypeS('DERChar', 'Char');
16729:   CL.AddTypeS('TEElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16730:     + 'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16731:   CL.AddTypeS('TEElementsTypes', 'set of TEElementsType');
16732:   CL.AddTypeS('DERNString', 'String');
16733:   const DERDecimalSeparator', 'String').SetString( '.' );
16734:   const DERDefaultChars', 'String')(+@/%
_.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16735:   const DERDefaultChars', 'String').SetString( '/%-.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16736:   Function isValidWebSiteChar( aChar: Char ) : Boolean';
16737:   Function isValidWebMailChar( aChar : Char ) : Boolean';
16738:   Function isValidwebSite( aStr : String ) : Boolean';
16739:   Function isValidWebMail( aStr : String ) : Boolean';
16740:   Function ValidateDate( aDERStr : DERString; var RsltFormat : String ) : Boolean';
16741:   Function DERStrToDate( aDERStr, aFormat : String ) : TDate';
16742:   Function IsDERChar( aChar : Char ) : Boolean';
16743:   Function IsDERDefaultChar( aChar : Char ) : Boolean';
16744:   Function IsDERMoneyChar( aChar : Char ) : Boolean';
16745:   Function IsDERExceptionCar( aChar : Char ) : Boolean';
16746:   Function IsDERSymbols( aDERString : String ) : Boolean';
16747:   Function StringToDERCharSet( aStr : String ) : DERString';
16748:   Function IsDERNDefaultChar( aChar : Char ) : Boolean';
16749:   Function IsDERNChar( aChar : Char ) : Boolean';
16750:   Function DERToDERCharset( aDERStr : DERString ) : DERNString';
16751:   Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean ) : String';
16752:   Function DERExtractWebMail( aDERStr : DERString ) : String';
16753:   Function DERExtractPhoneNr( aDERStr : DERString ) : String';
16754:   Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
SmartWebsiteRecognition:Boolean):TElementsType;
16755:   Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
SmartWebsiteRecognition:Boolean):TElementsType;
16756:   Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
aElementsType : TEElementsType ) : String';
16757:   Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
Boolean; var RsltDERStr : DERString; var RsltElementType : TEElementsType );';
16758: end;
16759:
16760: procedure SIRегистер_cyImage(CL: TPSPascalCompiler);
16761: begin
16762:   pRGBQuadArray', '^TRGBQuadArray // will not work';
16763:   Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer';
16764:   Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean );
16765:   Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean );
16766:   Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean );
16767:   Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean );

```

```

16768: Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)'');
16769: Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16770: Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean)');'
16771: Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool');'
16772: Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
SingleDestinationColor, RefreshBmp : Boolean)');'
16773: Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');'
16774: Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');'
16775: Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16776: Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16777: Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean)');'
16778: Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16779: Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');'
16780: Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');'
16781: end';
16782:
16783: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16784: begin
16785:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16786:   TPCMChannel', '( cMono, cStereo )');
16787:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16788:   TPCMBytesPerSample', '( bs8Bit, bs16Bit )');
16789:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono16b'
16790:     +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16791:     +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16792:     +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16793:     +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16794:     +'it48000Hz, Stereo16bit48000Hz )');
16795:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16796:     +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word; ecbSiz: Word; end');
16797:   tWaveFormatEx', 'PWaveFormatEx');
16798:   HMMIO', 'Integer');
16799:   TWaveDeviceFormats', 'set of TPCMFormat');
16800:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16801:     +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16802:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16803:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16804:   TWaveOutOptions', 'set of TWaveOutOption');
16805:   TStreamOwnership2', '( soReference, soOwned )');
16806:   TWaveStreamState', '( wsReady, wssReading, wssWriting, wssWritingEx )');
16807: // PRawWave', '^TRawWave // will not work');
16808: TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16809: CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioError');
16810: CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioSysError');
16811: CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioInvalidOperation');
16812: TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16813: TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16814:   +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16815: TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16816:   +'fee : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16817: TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16818:   +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16819: TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16820:   +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16821: TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16822: TWaveAudioFilterEvent', 'Procedure ( Sender : TObject; const Buffer:TObject; BufferSize:DWORD)');
16823: GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16824: CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16825: CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16826: GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16827: CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16828: OpenStreamWaveAudio( Stream : TStream ) : HMMIO');
16829: CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD ) : DWORD');
16830: GetAudioFormat( FormatTag : Word ) : String');
16831: GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx ) : String');
16832: GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD ) : DWORD');
16833: GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx ) : DWORD');
16834: GetWaveAudioPeakLevel( const Data : TObject; DataSize:DWORD; const pWaveFormat:PWaveFormatEx) : Integer');
16835: InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx) : Boolean');
16836: SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx) : Boolean');
16837: ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16838: MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
TObject; BufferSize : DWORD) : Boolean');
16839: ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16840: SetPCMFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
TPCMSamplesPerSec; BitsPerSample : TPCMBytesPerSample)');
16841: Procedure SetPCMFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat ) );
16842: GetPCMFormat( const pWaveFormat : PWaveFormatEx ) : TPCMFormat );
16843: GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD ) : DWORD );
16844: MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat ) : String );
16845: WaitForSyncObject( SyncObject : THandle; Timeout : DWORD ) : DWORD );
16846: //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD ) : LRESULT );
16847: end;
16848:
16849: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);

```

```

16850: begin
16851:   'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16852:   CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16853:   'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
16854:   'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ) );
16855:   'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
16856:   'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
16857:   'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
16858:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16859:   CL.AddTypeS('TPipeType', '( ptyp_ByByte, ptyp_MsgByte, ptyp_MsgMsg )');
16860:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16861:   SIRegister_TNamedPipe(CL);
16862:   SIRegister_TServePipe(CL);
16863:   SIRegister_TCClientPipe(CL);
16864:   Function CalculateTimeout( aBasis : DWORD ) : DWORD';
16865:   Function HasOverlapedIoCompleted( const ov : OVERLAPPED ) : Boolean';
16866:   Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean): TOverlappedResult;
16867:   Function GetStreamAsText( stm : TStream ) : string';
16868:   Procedure SetStreamAsText( const aTxt : string; stm : TStream )';
16869: end;
16870:
16871: procedure SIRegister_DPUtils(CL: TPSPPascalCompiler);
16872: begin
16873: // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16874:   SIRegister_TThumbData(CL);
16875:   'PIC_BMD','LongInt').SetInt( 0);
16876:   'PIC_JPG','LongInt').SetInt( 1);
16877:   'THUMB_WIDTH','LongInt').SetInt( 60);
16878:   'THUMB_HEIGHT','LongInt').SetInt( 60);
16879:   Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean';
16880:   Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)';
16881:   Function ReadBitmap( Filehandle : Integer; Width, Height : Integer ) : TBitmap';
16882:   Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap )';
16883:   Function OpenPicture( fn : string; var tp : Integer ) : Integer';
16884:   Function ConvertPicture( pi : Integer; tp : Integer ) : TBitmap';
16885:   Function LoadPicture( fn : string; var w, h : Integer ) : TBitmap';
16886:   Function TurnBitmap( bmp : TBitmap; ang : Integer ) : TBitmap';
16887:   Function RotateBitmap( Bitmap : TBitmap; Direction : Integer ) : TBitmap';
16888:   Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap ) : TRect';
16889:   Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer ) : TBitmap';
16890:   Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer )';
16891:   Procedure FindFiles( path, mask : string; items : TStringList )';
16892:   Function LetFileName( s : string ) : string';
16893:   Function LetParentPath( path : string ) : string';
16894:   Function AddBackSlash( path : string ) : string';
16895:   Function CutBackSlash( path : string ) : string';
16896: end;
16897:
16898: procedure SIRegister_CommonTools(CL: TPSPPascalCompiler);
16899: begin
16900: //BYTES','LongInt').SetInt( 1);
16901:   'KBYTES','LongInt').SetInt( 1024);
16902:   'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ) );
16903:   'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ) );
16904:   'DBG_GONE','LongWord').SetUInt( $99AC1D99);
16905:   'SHELL_NS_MYCOMPUTER','String').SetString( ':{20D04FE0-3AEA-1069-A2D8-08002B30309D} );
16906:   SIRegister_MakeComServerMethodsPublic(CL);
16907:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16908:   Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD ) : Boolean';
16909:   Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean )';
16910:   Function TBGetTempFolder : string';
16911:   Function TBGetTempFile : string';
16912:   Function TBGetModuleFilename : string';
16913:   Function FormatModuleVersionInfo( const aFilename : string ) : string';
16914:   Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD ) : string';
16915:   Function TBGetFileSize( aFile : string; aMultipleOf : Integer ) : Integer';
16916:   Function FormatAttribString( aAttr : Integer ) : string';
16917:   Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo ) : string';
16918:   Function ShellRecycle( aWnd : HWND; aFileOrFolder : string ) : Boolean';
16919:   Function IsDebuggerPresent : BOOL';
16920:   Function TBNotImplemented : HRESULT';
16921: end;
16922:
16923: procedure SIRegister_D2_VistaHelperU(CL: TPSPPascalCompiler);
16924: begin
16925: //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16926: //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16927: CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16928: CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean');
16929: //TDrivesProperty = array['A'..'Z'] of boolean;
16930: Function TBSetSystemTime( DateTime : TDateTime; DOW : word ) : boolean';
16931: Function IsElevated : Boolean';
16932: Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:GUID;const aIID:GUID;out aObj:TObject);
16933: CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu Defined )');
16934: Function TrimNetResource( UNC : string ) : string';
16935: Procedure GetFreeDrives( var FreeDrives : TDrivesProperty );
16936: Procedure GetMappedDrives( var MappedDrives : TDrivesProperty );
16937: Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
UserUsage : TPasswordUsage; User : string; Comment : string ) : boolean';

```

```

16938: Function UnmapDrive( Drive : char; Force : boolean) : boolean';
16939: Function TBIsWindowsVista : Boolean';
16940: Procedure SetVistaFonts( const AForm : TForm)');
16941: Procedure SetVistaContentFonts( const AFont : TFont)');
16942: Function GetProductType( var sType : String) : Boolean';
16943: Function lstrcmp( lpString1, lpString2 : PChar) : Integer';
16944: Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16945: Function lstrcpy( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar';
16946: Function lstrcpy( lpString1, lpString2 : PChar) : PChar';
16947: Function lstrcat( lpString1, lpString2 : PChar) : PChar';
16948: Function lstrlen( lpString : PChar) : Integer');
16949: Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
TokenInformation : __Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL';
16950: Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
TokenInformation : __Pointer; TokenInformationLength : DWORD) : BOOL';
16951: end;
16952:
16953: procedure SIRегистre_dwsXPlatform(CL: TPPSPascalCompiler);
16954: begin
16955:   'cLineTerminator','Char').SetString( #10);
16956:   'cLineTerminators','String').SetString( #13#10);
16957:   'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD( - 1 ) );
16958:   SIRегистre_TFixedCriticalSection(CL);
16959:   SIRегистre_TMultiReadSingleWrite(CL);
16960:   Procedure SetDecimalSeparator( c : Char)';
16961:   Function GetDecimalSeparator : Char';
16962:   TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16963:   Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16964:   CL.AddTypeS('NativeInt', 'Integer');
16965:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16966:   CL.AddTypeS('NativeUIInt', 'Cardinal');
16967:   //CL.AddTypeS('PNativeUIInt', '^NativeUIInt // will not work');
16968:   //CL.AddTypeS('TBytes', 'array of Byte');
16969:   CL.AddTypeS('RawByteString', 'UnicodeString');
16970:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16971:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16972:   SIRегистre_TPath(CL);
16973:   SIRегистre_TFile(CL);
16974:   SIRегистre_TDwsThread(CL);
16975:   Function GetSystemMilliseconds : Int64';
16976:   Function UTCDateTime : TDateTime';
16977:   Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString';
16978:   Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer';
16979:   Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer';
16980:   Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer';
16981:   Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer';
16982:   Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer';
16983:   Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString';
16984:   Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString';
16985:   Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer';
16986:   Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean';
16987:   Function InterlockedIncrement( var val : Integer) : Integer';
16988:   Function InterlockedDecrement( var val : Integer) : Integer';
16989:   Procedure FastInterlockedIncrement( var val : Integer)');
16990:   Procedure FastInterlockedDecrement( var val : Integer)');
16991:   Function InterlockedExchangePointer( var target : __Pointer; val : __Pointer) : __Pointer';
16992:   Procedure SetThreadName( const threadName : Char; threadID : Cardinal)';
16993:   Procedure dwsOutputDebugString( const msg : UnicodeString)';
16994:   Procedure WriteToOSEventLog( const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16995:   Function TryTextToFloat( const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16996:   Procedure VarCopy( out dest : Variant; const src : Variant)');
16997:   Function VarToUnicodeStr( const v : Variant) : UnicodeString';
16998:   Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString';
16999:   Function LoadTextFromStream( aStream : TStream) : UnicodeString';
17000:   Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString';
17001:   Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)';
17002:   Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle';
17003:   Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle';
17004:   Procedure CloseFileHandle( hFile : THandle)');
17005:   Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean';
17006:   Function FileMove( const existing, new : UnicodeString) : Boolean';
17007:   Function dwsFileDelete( const fileName : String) : Boolean';
17008:   Function FileRename( const oldName, newName : String) : Boolean';
17009:   Function dwsFileSize( const name : String) : Int64';
17010:   Function dwsFileDateTime( const name : String) : TDateTime';
17011:   Function DirectSet8087CW( newValue : Word) : Word';
17012:   Function DirectSetMXCSR( newValue : Word) : Word';
17013:   Function TtoObject( const T: byte) : TObject';
17014:   Function TtoPointer( const T: byte) : __Pointer';
17015:   Procedure GetMemForT(var T: byte; Size : integer)';
17016:   Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');
17017: end;
17018:
17019: procedure SIRегистre_AdSocket(CL: TPPSPascalCompiler);
17020: begin
17021:   'IPStrSize','LongInt').SetInt( 15 );
17022:   'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711 );
17023:   'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712 );

```

```

17024:   'ADWSBASE', 'LongInt').SetInt( 9000);
17025:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
17026:     +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
17027:   SIRegister_TApdSocketException(CL);
17028:   TWsMode', '( wsClient, wsServer )');
17029:   TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket)');
17030:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrorCode : Integer)');
17031:   SIRegister_TApdSocket(CL);
17032: end;
17033:
17034: procedure SIRegister_AdPort(CL: TPSPPascalCompiler);
17035: begin
17036:   SIRegister_TApdCustomComPort(CL);
17037:   SIRegister_TApdComPort(CL);
17038:   Function ComName( const ComNumber : Word ) : ShortString';
17039:   Function SearchComPort( const C : TComponent ) : TApdCustomComPort';
17040: end;
17041:
17042: procedure SIRegister_PathFunc(CL: TPSPPascalCompiler);
17043: begin
17044:   Function inAddBackslash( const S : String ) : String';
17045:   Function PathChangeExt( const Filename, Extension : String ) : String';
17046:   Function PathCharCompare( const S1, S2 : PChar ) : Boolean';
17047:   Function PathCharIsSlash( const C : Char ) : Boolean';
17048:   Function PathCharIsTrailByte( const S : String; const Index : Integer ) : Boolean';
17049:   Function PathCharLength( const S : String; const Index : Integer ) : Integer';
17050:   Function inPathCombine( const Dir, Filename : String ) : String';
17051:   Function PathCompare( const S1, S2 : String ) : Integer';
17052:   Function PathDrivePartLength( const Filename : String ) : Integer';
17053:   Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;
17054:   Function inPathExpand( const Filename : String ) : String';
17055:   Function PathExtensionPos( const Filename : String ) : Integer';
17056:   Function PathExtractDir( const Filename : String ) : String';
17057:   Function PathExtractDrive( const Filename : String ) : String';
17058:   Function PathExtractExt( const Filename : String ) : String';
17059:   Function PathExtractName( const Filename : String ) : String';
17060:   Function PathExtractPath( constFilename : String ) : String';
17061:   Function PathIsRooted( const Filename : String ) : Boolean';
17062:   Function PathLastChar( const S : String ) : PChar';
17063:   Function PathLastDelimiter( const Delimiters, S : string ) : Integer';
17064:   Function PathLowercase( const S : String ) : String';
17065:   Function PathNormalizeSlashes( const S : String ) : String';
17066:   Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
17067:   Function PathPos( Ch : Char; const S : String ) : Integer';
17068:   Function PathStartsWith( const S, AStartsWith : String ) : Boolean';
17069:   Function PathStrNextChar( const S : PChar ) : PChar';
17070:   Function PathStrPrevChar( const Start, Current : PChar ) : PChar';
17071:   Function PathStrScan( const S : PChar; const C : Char ) : PChar';
17072:   Function inRemoveBackslash( const S : String ) : String';
17073:   Function RemoveBackslashUnlessRoot( const S : String ) : String';
17074:   Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean );
17075: end;
17076:
17077:
17078: procedure SIRegister_CmnFunc2(CL: TPSPPascalCompiler);
17079: begin
17080:   NEWREGSTR_PATH_SETUP', 'String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17081:   NEWREGSTR_PATH_EXPLORER', 'String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17082:   NEWREGSTR_PATH_SPECIAL_FOLDERS', 'String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17083:   NEWREGSTR_PATH_UNINSTALL', 'String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17084:   NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME', 'String').SetString( 'DisplayName');
17085:   NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE', 'String').SetString( 'UninstallString');
17086:   KEY_WOW64_64KEY', 'LongWord').SetUInt( $0100);
17087:   //CL.AddTypeS('TLeadByteSet', 'TLeadByteSet // will not work');
17088:   CL.AddTypeS('TLeadByteSet', 'set of Char');
17089:   SIRegister_TOneShotTimer(CL);
17090:   CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17091:   'RegViews64Bit', 'LongInt').Value.ts32 := ord(rv64bit);
17092:   Function NewFileExists( const Name : String ) : Boolean';
17093:   Function inDirExists( const Name : String ) : Boolean';
17094:   Function FileOrDirExists( const Name : String ) : Boolean';
17095:   Function IsDirectoryAndNotReparsePoint( const Name : String ) : Boolean';
17096:   Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String';
17097:   Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17098:   Function GetIniBool( const Section,Key:String; const Default:Boolean;const Filenam:String): Boolean';
17099:   Function IniKeyExists( const Section, Key, Filename : String ) : Boolean';
17100:   Function IsIniSectionEmpty( const Section, Filename : String ) : Boolean';
17101:   Function SetIniString( const Section, Key, Value, Filename : String ) : Boolean';
17102:   Function SetIni( const Section,Key:String;const Value: Longint;const Filenam: String):Boolean';
17103:   Function SetIniBool( const Section,Key:String;const Value: Boolean;const Filenam: String):Boolean';
17104:   Procedure DeleteIniEntry( const Section, Key, Filename : String );
17105:   Procedure DeleteIniSection( const Section, Filename : String );
17106:   Function GetEnv( const EnvVar : String ) : String';
17107:   Function GetCmdTail : String';
17108:   Function GetCmdTailEx( StartIndex : Integer ) : String';
17109:   Function NewParamCount : Integer';
17110:   Function NewParamStr( Index : Integer ) : string';
17111:   Function AddQuotes( const S : String ) : String';
17112:   Function RemoveQuotes( const S : String ) : String';

```

```

17113: Function inGetShortName( const LongName : String ) : String';
17114: Function inGetWinDir : String');
17115: Function inGetSystemDir : String');
17116: Function GetSysWow64Dir : String');
17117: Function GetSysNativeDir( const IsWin64 : Boolean ) : String');
17118: Function inGetTempDir : String');
17119: Function StringChange( var S : String; const FromStr,ToStr : String ) : Integer');
17120: Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17121: Function AdjustLength( var S : String; const Res : Cardinal ) : Boolean');
17122: Function UsingWinNT : Boolean');
17123: Function ConvertConstPercentStr( var S : String ) : Boolean');
17124: Function ConvertPercentStr( var S : String ) : Boolean');
17125: Function ConstPos( const Ch : Char; const S : String ) : Integer');
17126: Function SkipPastConst( const S : String; const Start : Integer ) : Integer');
17127: Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String ) : Boolean');
17128: Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String ):Boolean;
17129: Function RegValueExists( H : HKEY; Name : PChar ) : Boolean');
17130: Function RegCreateKeyExView(const RegView:TRegView;hKey:HKEY;lpSubKey:PChar;Reserved:DWORD;lpClass:PChar;
dwOptions:DWORD;samDesired:REGSAM;lpSecurityAttributes:TObject;var
phkResult:HKEY;lpdwDisposition:DWORD):Longint;
17131: Function RegOpenKeyExView(const
RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17132: Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar ) : Longint;
17133: Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17134: Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyName:PChar):Longint;
17135: Function GetShellFolderPath( const FolderID : Integer ) : String');
17136: Function IsAdminLoggedOn : Boolean');
17137: Function IsPowerUserLoggedOn : Boolean');
17138: Function IsMultiByteString( const S : AnsiString ) : Boolean');
17139: Function FontExists( const FaceName : String ) : Boolean');
17140: //Procedure FreeAndNil( var Obj );
17141: Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT ) : HMODULE');
17142: Function GetUILanguage : LANGID');
17143: Function RemoveAccelChar( const S : String ) : String');
17144: Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17145: Function AddPeriod( const S : String ) : String');
17146: Function GetExceptMessage : String');
17147: Function GetPreferredUIFont : String');
17148: Function IsWildcard( const Pattern : String ) : Boolean');
17149: Function WildcardMatch( const Text, Pattern : PChar ) : Boolean');
17150: Function IntMax( const A, B : Integer ) : Integer');
17151: Function Win32ErrorString( ErrorCode : Integer ) : String');
17152: Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet );
17153: Function inCompareMem( P1, P2 : TObject; Length : Integer ) : Boolean');
17154: Function DeleteDirTree( const Dir : String ) : Boolean');
17155: Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
17156: Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT );
17157: // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17158: Function inCharInSet( C : Char; const CharSet : TSysCharSet ) : Boolean');
17159: Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String ) : Boolean');
17160: Function ShutdownBlockReasonDestroy( Wnd : HWND ) : Boolean');
17161: Function TryStrToBoolean( const S : String; var BoolResult : Boolean ) : Boolean');
17162: Procedure WaitMessageWithTimeout( const Milliseconds : DWORD );
17163: Function MoveFileReplace(const ExistingFileName, NewFileName : String ) : Boolean');
17164: Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND );
17165: end;
17166:
17167: procedure SIRegister_CmnFunc(CL: TPPascalCompiler);
17168: begin
17169:   SIRegister_TWindowDisabler(CL);
17170:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17171:   TMsgBoxCallbackFunc', 'procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17172:   Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox );
17173:   Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17174:   Function AppMessageBox( const Text, Caption : PChar; Flags : Longint ) : Integer');
17175:   Function MsgBoxP( const Text,Caption: PChar; const Typ: TMMsgBoxType;const Buttons:Cardinal):Int;
17176:   Function inMsgBox(const Text,Caption:String; const Typ: TMMsgBoxType;const Buttons:Cardinal):Int;
17177:   Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
Typ:TMMsgBoxType;const Buttons:Cardinal):Integer );
17178:   Procedure ReactivateTopWindow );
17179:   Procedure SetMessageBoxCaption( const Typ : TMMsgBoxType; const NewCaption : PChar );
17180:   Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean );
17181:   Procedure SetMessageBoxCallbackFunc(const AFunc : TMMsgBoxCallbackFunc; const AParam : LongInt );
17182: end;
17183:
17184: procedure SIRegister_ImageGrabber(CL: TPPascalCompiler);
17185: begin
17186:   SIRegister_TImageGrabber(CL);
17187:   SIRegister_TCaptureDrivers(CL);
17188:   SIRegister_TCaptureDriver(CL);
17189: end;
17190:
17191: procedure SIRegister_SecurityFunc(CL: TPPascalCompiler);
17192: begin
17193:   Function GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:String;const
Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean );
17194:   Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean );
17195: end;

```

```

17196:
17197: procedure SIRегистер_RedirFunc(CL: TPSPascalCompiler);
17198: begin
17199:   CL.AddTypeS('TPreviousFsRedirectionState', 'record DidDisable : Boolean; OldValue : __Pointer; end');
17200:   Function AreFsRedirectionFunctionsAvailable : Boolean';
17201:   Function DisableFsRedirectionIf(const Disable : Boolean; var PreviousState:TPreviousFsRedirectionState):Bool;
17202:   Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState );
17203:   Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String ) : BOOL';
17204:   Function CreateProcessRedir( const DisableFsRedir: Boolean; const lpApplicationName:PChar; const
lpCommandLine:PChar; const lpProcessAttributes, lpThreadAttributes:PSecurityAttributes; const
bInheritHandles:BOOL; const dwCreationFlags:DWORD; const lpEnvironment:Pointer; const
lpCurrentDirectory:PChar; const lpStartupInfo : TStartupInfo; var
lpProcessInformation:TProcessInformation):BOOL';
17205:   Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
FailIfExists : BOOL) : BOOL';
17206:   Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String ) : BOOL';
17207:   Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String ) : Boolean';
17208:   Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String ) : Boolean';
17209:   Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
TWin32FindData ) : THandle';
17210:   Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String ) : DWORD';
17211:   Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String ) : String';
17212:   Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
TFileVersionNumbers ) : Boolean';
17213:   Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17214:   Function MoveFileRedir( const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
17215:   Function MoveFileExRedir( const DisableFsRedir:Bool;const ExistingFilen,NewFilename:String;const
Flags:DWORD ):BOOL;
17216:   Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String ) : Boolean';
17217:   Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String ) : BOOL';
17218:   Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17219:   Function SetNTFSCompressionRedir( const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17220:     SIRегистер_TFileRedir(CL);
17221:     SIRегистер_TTextFileReaderRedir(CL);
17222:     SIRегистер_TTextFileWriterRedir(CL);
17223:   end;
17224:
17225: procedure SIRегистер_Int64Em(CL: TPSPascalCompiler);
17226: begin
17227:   //CL.AddTypeS('LongWord', 'Cardinal');
17228:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17229:   Function Compare64( const N1, N2 : Integer64 ) : Integer';
17230:   Procedure Dec64( var X : Integer64; N : LongWord );
17231:   Procedure Dec6464( var X : Integer64; const N : Integer64 );
17232:   Function Div64( var X : Integer64; const Divisor : LongWord ) : LongWord';
17233:   Function Inc64( var X : Integer64; N : LongWord ) : Boolean';
17234:   Function Inc6464( var X : Integer64; const N : Integer64 ) : Boolean';
17235:   Function Integer64ToStr( X : Integer64 ) : String';
17236:   Function Mod64( const X : Integer64; const Divisor : LongWord ) : LongWord';
17237:   Function Mul64( var X : Integer64; N : LongWord ) : Boolean';
17238:   Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64 );
17239:   Procedure Shr64( var X : Integer64; Count : LongWord );
17240:   Function StrToInteger64( const S : String; var X : Integer64 ) : Boolean';
17241: end;
17242:
17243: procedure SIRегистер_InstFunc(CL: TPSPascalCompiler);
17244: begin
17245:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17246:   SIRегистер_TSsimpleStringList(CL);
17247:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle )');
17248:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17249:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17250:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17251:   // TMD5Digest = array[0..15] of Byte;
17252:   // TSHA1Digest = array[0..19] of Byte;
17253:   Function CheckForMutexes( Mutexes : String ) : Boolean';
17254:   Function CreateTempDir : String';
17255:   Function DecrementSharedCount( const RegView : TRegView; const Filename : String ) : Boolean';
17256:   Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
FirstRetryDelayMS, SubsequentRetryDelayMS : Integer );
17257:   //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
TDeleteFileProc; const Param : Pointer ) : Boolean';
17258:   //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer ) :
TDetermineDefaultLanguageResult';
17259:   //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17260:   Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String ) : Boolean';
17261:   Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17262:   Function GetComputerNameString : String';
17263:   Function GetFileDateTime(const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime):Bool;
17264:   Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String ) : TMD5Digest';
17265:   Function GetMD5OfAnsiString( const S : AnsiString ) : TMD5Digest';
17266:   // Function GetMD5OfUnicodeString( const S : UnicodeString ) : TMD5Digest';
17267:   Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest';
17268:   Function GetSHA1OfAnsiString( const S : AnsiString ) : TSHA1Digest';
17269:   // Function GetSHA1OfUnicodeString( const S : UnicodeString ) : TSHA1Digest';
17270:   Function GetRegRootKeyName( const RootKey : HKEY ) : String';
17271:   Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
TotBytes:Int64):Bool;

```

```

17272: Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
17273: TotalBytes: Integer64):Bool;
17274: Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17275: //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
17276: String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
17277: ResultCode:Integer) : Boolean';
17278: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
17279: TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17280: Procedure InternalError( const Id : String)');
17281: Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17282: Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean');
17283: Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17284: Function MakePendingFileRenameOperationsChecksum : TMD5Digest');
17285: Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17286: Procedure RaiseFunctionFailedError( const FunctionName : String)');
17287: Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17288: Procedure RefreshEnvironment ')';
17289: Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17290: Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17291: Procedure UnregisterFont( const FontName, FontFilename : String)');
17292: Function RestartComputer : Boolean');
17293: Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17294: Procedure SplitNewParamStr( const Index: Integer; var AName, AValue : String)');
17295: Procedure Win32ErrorMsg( const FunctionName : String)');
17296: Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17297: Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17298: //from Func2
17299: //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
17300: Iconfilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean;
17301: //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String';
17302: Procedure RegisterTypeLibrary( const Filename : String)');
17303: //Procedure UnregisterTypeLibrary( const Filename : String)');
17304: function getVersionInfoEx3: TOSVersionInfoEx)';
17305: Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17306: procedure InitOle();
17307: Function ExpandConst( const S : String) : String');
17308: Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17309: Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17310: Procedure ExpandConstIfPrefixed( const S : String) : String');
17311: Procedure LogWindowsVersion();
17312: Function EvalCheck( const Expression : String) : Boolean');
17313: begin
17314: CL.AddClassN(CL.FindClass('TOBJECT'), 'TResourceDetails');
17315: //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17316: SIRegister_TResourceModule(CL);
17317: SIRegister_TResourceDetails(CL);
17318: SIRegister_TAnsiResourceDetails(CL);
17319: SIRegister_TUnicodeResourceDetails(CL);
17320: Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17321: Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17322: Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17323: Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17324: Function ResourceNameToInt( const s : string) : Integer');
17325: Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer');
17326: end;
17327:
17328:
17329: procedure SIRegister_TSsimpleComPort(CL: TPPascalCompiler);
17330: begin
17331: //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17332: with CL.AddClassN(CL.FindClass('TObject'), 'TSimpleComPort') do begin
17333: RegisterMethod('Constructor Create');
17334: RegisterMethod('Procedure Free');
17335: RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17336: RegisterMethod('Procedure WriteString( const S : String)');
17337: RegisterMethod('Procedure ReadString( var S : String)');
17338: end;
17339: Ex:= SimpleComPort:= TSsimpleComPort.Create;
17340: SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17341: SimpleComPort.WriteString(AsciiChar);
17342: end;
17343:
17344:
17345: procedure SIRegister_Console(CL: TPPascalCompiler);
17346: begin
17347: CL.AddConstantN('White','LongInt').SetInt( 15); }
17348: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17349: ('conBW40','LongInt').SetInt( 0);
17350: ('conCO40','LongInt').SetInt( 1);
17351: ('conBW80','LongInt').SetInt( 2);
17352: ('conCO80','LongInt').SetInt( 3);
17353: ('conMono','LongInt').SetInt( 7);
17354: CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17355: //CL.AddConstantN('C40','').SetString( CO40);

```

```

17356: //CL.AddConstantN('C80','').SetString( C080);
17357: Function con.ReadKey : Char');
17358: Function conKeyPressed : Boolean');
17359: Procedure conGotoXY( X, Y : Smallint)');
17360: Function conWhereX : Integer');
17361: Function conWhereY : Integer');
17362: Procedure conTextColor( Color : Byte);');
17363: Function conTextColor1 : Byte;');
17364: Procedure conTextBackground( Color : Byte);');
17365: Function conTextBackground1 : Byte);');
17366: Procedure conTextMode( Mode : Word);');
17367: Procedure conLowVideo');
17368: Procedure conHighVideo');
17369: Procedure conNormVideo');
17370: Procedure conClrScr');
17371: Procedure conClrEol');
17372: Procedure conInsLine');
17373: Procedure conDelLine');
17374: Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17375: Function conScreenWidth : Smallint');
17376: Function conScreenHeight : Smallint');
17377: Function conBufferWidth : Smallint');
17378: Function conBufferHeight : Smallint');
17379: procedure InitScreenMode;');
17380: end;
17381:
17382: (*-----*)
17383: procedure SIRегистер_тестutils(CL: TPSPPascalCompiler);
17384: begin
17385:   SIRегистер_TNoRefCountObject(CL);
17386:   Procedure FreeObjects( List : TFPList)');
17387:   Procedure GetMethodList( AObject : TObject; AList : TStrings)');
17388:   Procedure GetMethodList1( AClass : TClass; AList : TStrings)');
17389: end;
17390:
17391: procedure SIRегистер_ToolsUnit(CL: TPSPPascalCompiler);
17392: begin
17393:   'MaxDataSet', 'LongInt').SetInt( 35);
17394:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17395:   SIRегистер_TDBConnector(CL);
17396:   SIRегистер_TDBBasicsTestSetup(CL);
17397:   SIRегистер_TTestDataLink(CL);
17398:   'testValuesCount','Longint').SetInt( 25);
17399:   Procedure InitialiseDBConnector');
17400:   Procedure FreeDBConnector');
17401:   Function DateTimeToString( d : tdatetime) : string');
17402:   Function TStringToDate( d : String) : TDate');
17403: end;
17404:
17405: procedure SIRегистер_fpcunit(CL: TPSPPascalCompiler);
17406: begin
17407:   SIRегистер_EAssertionFailedError(CL);
17408:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17409:   CL.AddTypeS('TRunMethod', 'Procedure');
17410:   CL.AddClass(CL.FindClass('TOBJECT'),'TTestResult');
17411:   SIRегистер_TTest(CL);
17412:   SIRегистер_TAssert(CL);
17413:   SIRегистер_TTestFailure(CL);
17414:   SIRегистер_ITestListener(CL);
17415:   SIRегистер_TTestCase(CL);
17416:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17417:   SIRегистер_TTestSuite(CL);
17418:   SIRегистер_TTestResult(CL);
17419:   Function ComparisonMsg( const aExpected : string; const aActual : string) : string');
17420: end;
17421:
17422: procedure SIRегистер_cTCPBuffer(CL: TPSPPascalCompiler);
17423: begin
17424:   TOBJECT','ETCPBuffer');
17425:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Integer';
17426:     +r; Head : Integer; Used : Integer; end');
17427:   'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
17428:   'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
17429:   'TCP_BUFFER_DEFAULTMAXSIZE','Longint').SetInt( ETHERNET_MTU_1GBT * 4);
17430:   'TCP_BUFFER_DEFAULTBUFSIZE','Longint').SetInt( ETHERNET_MTU_100MBIT * 4);
17431:   Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int;const TCPBufSize:Int);
17432:   Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer)');
17433:   Procedure TCPBufferPack( var TCPBuf : TTCPBuffer)');
17434:   Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer)');
17435:   Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer)');
17436:   Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer)');
17437:   Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer) : Pointer');
17438:   Procedure TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf : string; const Size : Integer)');
17439:   Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Integer');
17440:   Function TCPBufferPeek( var TCPBuf : TTCPBuffer; var Buf: string; const Size:Integer): Integer');
17441:   Function TCPBufferRemove( var TCPBuf : TTCPBuffer;var Buf: string; const Size:Integer): Integer');
17442:   Procedure TCPBufferClear( var TCPBuf : TTCPBuffer)');
17443:   Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer) : Integer');
17444:   Function TCPBufferUsed( const TCPBuf : TTCPBuffer) : Integer');

```

```

17445: Function TCPBufferEmpty( const TCPBuf : TTCPBuffer ) : Boolean';
17446: Function TCPBufferAvailable( const TCPBuf : TTCPBuffer ) : Integer';
17447: Function TCPBufferPtr( const TCPBuf : TTCPBuffer ) : Pointer';
17448: Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer );
17449: end;
17450:
17451: procedure SIRegister_Glut(CL: TPSPascalCompiler);
17452: begin
17453: //CL.AddTypeS('PInteger', '^Integer // will not work');
17454: //CL.AddTypeS('PPChar', '^PChar // will not work');
17455: CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3 );
17456: CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12 );
17457: CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0 );
17458: CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5 );
17459: CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6 );
17460: Procedure LoadGlut( const dll : String );
17461: Procedure FreeGlut();
17462: end;
17463:
17464: procedure SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
17465: begin
17466: CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17467: Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean ) : THandle';
17468: end;
17469:
17470: procedure SIRegister_SwitchLed(CL: TPSPascalCompiler);
17471: begin
17472: TLedColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black )';
17473: TTLedState', '( LedOn, LedOff, LedDisabled )';
17474: SIRegister_TSwitchLed(CL);
17475: //Procedure Register';
17476: end;
17477:
17478: procedure SIRegister_FileClass(CL: TPSPascalCompiler);
17479: begin
17480: CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17481: +'xisting, fdOpenAlways, fdTruncateExisting )');
17482: CL.AddTypeS('TFileAccess', '( faRead, faWrite, faReadWrite )');
17483: CL.AddTypeS('TFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17484: SIRegister_TCustomFile(CL);
17485: SIRegister_TIFile(CL);
17486: SIRegister_TMemoryFile(CL);
17487: SIRegister_TTextFileReader(CL);
17488: SIRegister_TTextFileWriter(CL);
17489: SIRegister_TFileMapping(CL);
17490: SIRegister_EFileError(CL);
17491: end;
17492:
17493: procedure SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
17494: begin
17495: CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17496: +' , ffaDirectory, ffaArchive, ffaAnyFile )');
17497: CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17498: SIRegister_TFileSearch(CL);
17499: end;
17500:
17501: procedure SIRegister_uColorFunctions(CL: TPSPascalCompiler);
17502: begin
17503: TRGBTType', 'record RedHex : string; GreenHex : string; BlueHex :'
17504: +' string; Red : integer; Green : integer; Blue : integer; end');
17505: Function FadeColor( aColor : Longint; aFade : integer ) : Tcolor';
17506: Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17507: end;
17508:
17509: procedure SIRegister_uSettings(CL: TPSPascalCompiler);
17510: begin
17511: Procedure SaveOscSettings();
17512: Procedure GetOscSettings();
17513: end;
17514:
17515: procedure SIRegister_cyDebug(CL: TPSPascalCompiler);
17516: begin
17517: TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer )';
17518: RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17519: 'FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17520: '64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17521: 'Cardinal; EnterCount : Integer; ExitCount : Integer; end';
17522: SIRegister_TcyDebug(CL);
17523: end;
17524:
17525: (*-----*)
17526: procedure SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
17527: begin
17528: TCopyFileResult', '( cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )';
17529: TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )';
17530: TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )';
17531: SIRegister_TdestinationOptions(CL);
17532: TProcCustomCopyFileEvent', 'Procedure ( Sender : TObject; var CopyFileResult : TCopyFileResult )';
17533: TProcOnCopyFileProgressEvent', 'Procedure(Sender:TObject;FileBytes,
TransferredBytes:int64;PercentDone:Int64);
```

```

17534: TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String)');
17535: SIRegister_TcyCopyFiles(CL);
17536: Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
ResetAttr:boolean): TCopyFileResult');
17537: Function cyCopyFileEx(FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult');
17538: Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;' +
FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer');
17539: end;
17540: 
17541: 
17542: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17543: begin
17544:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17545:   SIRegister_TcyFileAttributes(CL);
17546:   SIRegister_TSearchRecInstance(CL);
17547:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17548:   TOptions', 'set of TOption');
17549:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17550:   TProcOnValidateFileEvent Procedure(Sender: TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
Accept:bool;
17551:   TProcOnValidateDirectoryEvent', 'Procedure(Sender:TObject;Directory:String/var Accept:boolean)');
17552:   SIRegister_TcyCustomSearchfiles(CL);
17553:   SIRegister_TcySearchFiles(CL);
17554:   Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean');
17555:   Function IscyFolder( aSRec : TSearchrec ) : Boolean');
17556: end;
17557: 
17558: procedure SIRegister_jcontrolutils(CL: TPSPascalCompiler);
17559: begin
17560:   Function jCountChar( const s : string; ch : char ) : integer');
17561:   Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings );
17562:   Function jNormalizeDate(const Value: string; theValue: TDateTime;const theFormat:string): string');
17563:   Function jNormalizeTime(const Value: string; theValue: TTime;const theFormat : string) : string');
17564:   Function jNormalizeDateTime(const Value:string;theValue:TDateTime;const theFormat:string):string');
17565:   Function jNormalizeDateSeparator( const s : string ) : string');
17566:   Function jIsValidDateString( const Value : string ) : boolean');
17567:   Function jIsValidTimeString( const Value : string ) : boolean');
17568:   Function jIsValidDateTimeString( const Value : string ) : boolean');
17569: end;
17570: 
17571: procedure SIRegister_kcMapView(CL: TPSPascalCompiler);
17572: begin
17573:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMapView');
17574:   CL.AddTypeS('TMapView', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo' +
'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik' +
', msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual' +
'+EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa' +
'hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17575:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17576:   TRealArea', 'record top : Extended; left : Extended; bottom : Extended; right : Extended; end');
17577:   TIntPoint', 'record X : Int64; Y : Int64; end');
17578:   TkcRealPoint', 'record X : Extended; Y : Extended; end');
17579:   TOnBeforeDownloadEvent', 'Procedure ( Url : string; str : TStream; var CanHandle : Boolean)');
17580:   TOnAfterDownloadEvent', 'Procedure ( Url : string; str : TStream)');
17581:   SIRegister_TCustomeDownloadEngine(CL);
17582:   SIRegister_TCustomGeolocationEngine(CL);
17583:   SIRegister_TMapView(CL);
17584: end;
17585: 
17586: procedure SIRegister_cparserutils(CL: TPSPascalCompiler);
17587: begin
17588:   (*Function isFunc( name : TNamePart ) : Boolean');*)
17589:   Function isUnnamedFunc( name : TNamepart ) : Boolean');
17590:   Function isPtrToFunc( name : TNamePart ) : Boolean');
17591:   Function isFuncRetFuncPtr( name : TNamePart ) : Boolean');
17592:   Function isPtrToFuncRetFuncPtr( name : TNamePart ) : Boolean');
17593:   Function GetFuncParam( name : TNamePart ) : TNamePart');
17594:   Function isArray( name : TNamePart ) : Boolean');
17595:   Function GetArrayPart( name : TNamePart ) : TNamePart');
17596:   Function GetIdFromPart( name : TNamePart ) : AnsiString');
17597:   Function GetIdPart( name : TNamePart ) : TNamePart');
17598:   Function isNamePartPtrToFunc( part : TNamePart ) : Boolean');
17599:   Function isAnyBlock( part : TNamePart ) : Boolean');*/
17600:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17601:   SIRegister_TLineBreaker(CL);
17602:   CL.AddTypeS('TNameKind', 'Integer');
17603:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNamePart');
17604:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17605:   Function SphericalMod( X : Extended ) : Extended');
17606:   Function cSign( Value : Extended ) : Extended');
17607:   Function LimitFloat( const eValue, eMin, eMax : Extended ) : Extended');
17608:   Function AngleToRadians( iAngle : Extended ) : Extended');
17609:   Function RadiansToAngle( eRad : Extended ) : Extended');
17610:   Function Cross180( iLong : Double ) : Boolean');
17611:   Function Mod180( Value : integer ) : Integer');
17612:   Function Mod180Float( Value : Extended ) : Extended');
17613:   Function MulDivFloat( a, b, d : Extended ) : Extended');
17614:   Function LongDiff( iLong1, iLong2 : Double ) : Double');
17615: 
```

```

17619: Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap)');
17620: Function Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap');
17621: Function FixFilePath( const Inpath, CheckPath : string) : string');
17622: Function UnFixFilePath( const Inpath, CheckPath : string) : string');
17623: Procedure FillStringList( sl : TStringList; const aText : string)');
17624: end;
17625:
17626: procedure SIRegister_LedNumber(CL: TPSPPascalCompiler);
17627: begin
17628:   TLedSegmentSize', 'Integer');
17629:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised )');
17630:   SIRegister_TCustomLEDNumber(CL);
17631:   SIRegister_TLEDNumber(CL);
17632: end;
17633:
17634: procedure SIRegister_StStrL(CL: TPSPPascalCompiler);
17635: begin
17636:   CL.AddTypeS('LStrRec', 'record AllocSize : Longint; RefCount : Longint; Length : Longint; end');
17637:   CL.AddTypes('AnsiChar', 'Char');
17638:   CL.AddTypeS('BTable', 'array[0..255] of Byte'); //!!!
17639:   Function HexBL( B : Byte) : AnsiString';
17640:   Function HexWL( W : Word) : AnsiString';
17641:   Function HexLL( L : LongInt) : AnsiString';
17642:   Function HexPTRL( P : __Pointer) : AnsiString';
17643:   Function BinaryBL( B : Byte) : AnsiString';
17644:   Function BinaryWL( W : Word) : AnsiString';
17645:   Function BinaryLL( L : LongInt) : AnsiString';
17646:   Function OctalBL( B : Byte) : AnsiString';
17647:   Function OctalWL( W : Word) : AnsiString';
17648:   Function OctalLL( L : LongInt) : AnsiString';
17649:   Function Str2Int16L( const S : AnsiString; var I : SmallInt) : Boolean';
17650:   Function Str2WordL( const S : AnsiString; var I : Word) : Boolean';
17651:   Function Str2LongL( const S : AnsiString; var I : LongInt) : Boolean';
17652:   Function Str2RealL( const S : AnsiString; var R : Double) : Boolean';
17653:   Function Str2RealL( const S : AnsiString; var R : Real) : Boolean';
17654:   Function Str2ExtL( const S : AnsiString; var R : Extended) : Boolean';
17655:   Function Long2StrL( L : LongInt) : AnsiString';
17656:   Function Real2StrL( R : Double; Width : Byte; Places : ShortInt) : AnsiString';
17657:   Function Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : AnsiString';
17658:   Function ValPrepL( const S : AnsiString) : AnsiString';
17659:   Function CharStrL( C : Char; Len : Cardinal) : AnsiString';
17660:   Function PadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString';
17661:   Function PadLL( const S : AnsiString; Len : Cardinal) : AnsiString';
17662:   Function LeftPadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString';
17663:   Function LeftPadL( const S : AnsiString; Len : Cardinal) : AnsiString';
17664:   Function TrimLeadL( const S : AnsiString) : AnsiString';
17665:   Function TrimTrailL( const S : AnsiString) : AnsiString';
17666:   Function TrimL( const S : AnsiString) : AnsiString';
17667:   Function TrimSpacesL( const S : AnsiString) : AnsiString';
17668:   Function CenterChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString';
17669:   Function CenterL( const S : AnsiString; Len : Cardinal) : AnsiString';
17670:   Function EntabL( const S : AnsiString; TabSize : Byte) : AnsiString';
17671:   Function DetabL( const S : AnsiString; TabSize : Byte) : AnsiString';
17672:   Function ScrambleL( const S, Key : AnsiString) : AnsiString';
17673:   Function SubstituteL( const S, FromStr,ToStr : AnsiString) : AnsiString';
17674:   Function FilterL( const S, Filters : AnsiString) : AnsiString';
17675:   Function CharExistsL( const S : AnsiString; C : AnsiChar) : Boolean';
17676:   Function CharCountL( const S : AnsiString; C : AnsiChar) : Cardinal';
17677:   Function WordCountL( const S, WordDelims : AnsiString) : Cardinal';
17678:   Function WordPositionL( N : Cardinal; const S, WordDelims : AnsiString; var Pos : Cardinal) : Boolean';
17679:   Function ExtractWordL( N : Cardinal; const S, WordDelims : AnsiString) : AnsiString';
17680:   Function AsciiCountL( const S, WordDelims : AnsiString; Quote : AnsiChar) : Cardinal';
17681:   Function AsciiPositionL(N: Cardinal;const S,WordDelims:AnsiString;Quote:AnsiChar;var Pos:Cardinal):Bool;
17682:   Function ExtractAsciiL( N : Cardinal; const S, WordDelims : AnsiString; Quote : AnsiChar) : AnsiString';
17683:   Procedure WordWrapL(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17684:   Procedure WordWrap(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17685:   Function CompStringL( const S1, S2 : AnsiString) : Integer';
17686:   Function CompUCStringL( const S1, S2 : AnsiString) : Integer';
17687:   Function SoundexL( const S : AnsiString) : AnsiString';
17688:   Function MakeLetterSetL( const S : AnsiString) : Longint';
17689:   Procedure BMMakeTableL( const MatchString : AnsiString; var BT : BTable)');
17690:   Function BMSearchL(var Buffer,BufLength:Cardinal;var BT:BTable;const MatchString:AnsiString;var
Pos:Card):Bool;
17691:   Function BMSearchUCL( var Buffer, BufLength : Cardinal; var BT : BTable; const MatchString : AnsiString;
var Pos : Cardinal) : Boolean';
17692:   Function DefaultExtensionL( const Name, Ext : AnsiString) : AnsiString';
17693:   Function ForceExtensionL( const Name, Ext : AnsiString) : AnsiString';
17694:   Function JustFilenameL( const PathName : AnsiString) : AnsiString';
17695:   Function JustNameL( const PathName : AnsiString) : AnsiString';
17696:   Function JustExtensionL( const Name : AnsiString) : AnsiString';
17697:   Function JustPathnameL( const PathName : AnsiString) : AnsiString';
17698:   Function AddBackSlashL( const DirName : AnsiString) : AnsiString';
17699:   Function CleanPathNameL( const PathName : AnsiString) : AnsiString';
17700:   Function HasExtensionL( const Name : AnsiString; var DotPos : Cardinal) : Boolean';
17701:   Function CommaizeL( L : Longint) : AnsiString';
17702:   Function CommaizeChL( L : Longint; Ch : AnsiChar) : AnsiString';
17703:   Function FloatFormL(const Mask:AnsiString;R:TstFloat;const LtCurr:AnsiString;Sep,
DecPt:Char):AnsiString;
17704:   Function LongIntFormL(const Mask:AnsiString;L:LongInt;const LtCurr,
RtCurr:AnsiString;Sep:Char):AnsiString';

```

```

17705: Function StrChPosL( const P : AnsiString; C : AnsiChar; var Pos : Cardinal) : Boolean');
17706: Function StrStPosL( const P, S : AnsiString; var Pos : Cardinal) : Boolean');
17707: Function StrStCopyL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17708: Function StrChInsertL( const S : AnsiString; C : AnsiChar; Pos : Cardinal) : AnsiString');
17709: Function StrStInsertL( const S1, S2 : AnsiString; Pos : Cardinal) : AnsiString');
17710: Function StrChDeleteL( const S : AnsiString; Pos : Cardinal) : AnsiString');
17711: Function StrStDeleteL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17712: Function ContainsOnlyL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17713: Function ContainsOtherThanL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17714: Function CopyLeftL( const S : AnsiString; Len : Cardinal) : AnsiString');
17715: Function CopyMidL( const S : AnsiString; First, Len : Cardinal) : AnsiString');
17716: Function CopyRightL( const S : AnsiString; First : Cardinal) : AnsiString');
17717: Function CopyRightAbsL( const S : AnsiString; NumChars : Cardinal) : AnsiString');
17718: Function CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Cardinal;var
SubString:AString):Bool;
17719: Function CopyFromToWordL(const S,WordDelims,Word1,Word2:AnsiString;N1,N2:Cardinal;var
SubString:AnsiString):Bool;
17720: Function CopyWithinL( const S, Delimiter : AnsiString; Strip : Boolean) : AnsiString');
17721: Function DeleteFromNthWordL( const S, WordDelims : AnsiString; const AWord : AnsiString; N : Cardinal;
var SubString : AnsiString) : Boolean');
17722: Function DeleteFromToWordL( const S, WordDelims, Word1, Word2 : AnsiString; N1, N2 : Cardinal; var
SubString : AnsiString) : Boolean');
17723: Function DeleteWithinL( const S, Delimiter : AnsiString) : AnsiString');
17724: Function ExtractTokensL(const S,
Delims:AnsiString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Cardinal;
17725: Function IsChAlphaL( C : AnsiChar ) : Boolean');
17726: Function IsChNumericL( C : AnsiChar; const Numbers : AnsiString ) : Boolean');
17727: Function IsChAlphaNumericL( C : AnsiChar; const Numbers : AnsiString ) : Boolean');
17728: Function IsStrAlphaL( const S : AnsiString ) : Boolean');
17729: Function IsStrNumericL( const S, Numbers : AnsiString ) : Boolean');
17730: Function IsStrAlphaNumericL( const S, Numbers : AnsiString ) : Boolean');
17731: Function KeepCharsL( const S, Chars : AnsiString ) : AnsiString');
17732: Function LastWordL( const S, WordDelims, AWord : AnsiString; var Position : Cardinal ) : Boolean');
17733: Function LastWordAbsL( const S, WordDelims : AnsiString; var Position : Cardinal ) : Boolean');
17734: Function LastStringL( const S, AString : AnsiString; var Position : Cardinal ) : Boolean');
17735: Function LeftTrimCharsL( const S, Chars : AnsiString ) : AnsiString');
17736: Function ReplaceWordL(const S, WordDelims,OldWord,NewWord:AnsiString;N:Card;var
Replacements:Card):AnsiString;
17737: Function ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:AnsiString;var
Replacements:Cardinal):AnsiString';
17738: Function ReplaceStringL(const S,OldString,NewString:AnsiString;N:Cardinal;var
Replacements:Cardinal):AnsiString;
17739: Function ReplaceStringAllL(const S,OldString,NewString:AnsiString; var Replacements:Cardinal):AnsiString;
17740: Function RepeatStringL(const RepeatString:AnsiString;var Repetitions:Cardinal;MaxLen:Cardinal):AnsiString;
17741: Function RightTrimCharsL( const S, Chars : AnsiString ) : AnsiString');
17742: Function StrWithinL( const S, SearchStr : string; Start : Cardinal; var Position : Cardinal ) : boolean');
17743: Function TrimCharsL( const S, Chars : AnsiString ) : AnsiString');
17744: Function WordPosL(const S,WordDelims,AWord: AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17745: Function WordPos(const S,WordDelims,AWord:AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17746: end;
17747:
17748: procedure SIRegister_pwnative_out(CL: TPSPascalCompiler);
17749: begin
17750: CL.AddConstantN('STDIN','LongInt').SetInt( 0 );
17751: ('STDOUT','LongInt').SetInt( 1 );
17752: ('STDERR','LongInt').SetInt( 2 );
17753: Procedure NativeWrite( s : astr );';
17754: Procedure NativeWrite1( PString : PChar );';
17755: Procedure NativeWrite2( Buffer : PChar; NumChars : Cardinal );';
17756: Procedure NativeWriteLn( s : astr );';
17757: Procedure NativeWriteLn1();';
17758: end;
17759:
17760: procedure SIRegister_synwrapl(CL: TPSPascalCompiler);
17761: begin
17762: CL.AddTypeS(TSynwInfo', 'record Err : byte; UrlHtml : ansistring; ErrResponse
17763: : integer; UltimateURL : ansistring; Headers : ansistring; end');
17764: CL.AddTypeS('TUrlInfo', 'record Err : byte; UltimateURL : string; end');
17765: Function GetHttpFile( const Url, UserAgent, Outfile : string; verbose : boolean): TSynwInfo;');
17766: Function GetHttpFile1( const Url, UserAgent, Outfile : string ) : TSynwInfo;');
17767: Function GetHttpFile2( const Url, Outfile : string ) : TSynwInfo;');
17768: Function GetHttpFile3( const Url, outfile : string; verbose : boolean ) : TSynwInfo;');
17769: Function GetHtm( const Url : string ) : string;');
17770: Function GetHtm1( const Url, UserAgent : string ) : string;');
17771: Function GetUrl( const Url : string; verbose : boolean ) : TSynwInfo;');
17772: Function GetUrl1( const Url, useragent : string ) : TSynwInfo;');
17773: Function GetUrl2( const Url : string ) : TSynwInfo;');
17774: Function GetUrl3( const Url : string; const http : THTTPSend; verbose : boolean): TUrlInfo;');
17775: Function GetUrl4( const Url : string; const http : THTTPSend ) : TUrlInfo;');
17776: Procedure StrToStream( s : String; strm : TMemoryStream );
17777: Function StrLoadStream( strm : TStream ) : String');
17778: end;
17779:
17780: procedure SIRegister_HTMLUtil(CL: TPSPascalCompiler);
17781: begin
17782: Function GetVal( const tag, attribname_ci : string ) : string');
17783: Function GetTagName( const Tag : string ) : string');
17784: Function GetUpTagName( const tag : string ) : string');
17785: Function GetNameValPair( const tag, attribname_ci : string ) : string');

```

```

17786: Function GetValFromNameVal( const namevalpair : string ) : string';
17787: Function GetNameValPair_cs( const tag, attribname : string ) : string';
17788: Function GetVal_JAMES( const tag, attribname_ci : string ) : string';
17789: Function GetNameValPair_JAMES( const tag, attribname_ci : string ) : string');
17790: Function CopyBuffer( StartIndex : PChar; Len : integer ) : string');
17791: Function Ucase( s : string ) : string');
17792: end;
17793:
17794: procedure SIRegister_pwmain(CL: TPSPascalCompiler);
17795: begin
17796:   CL.AddConstantN('FUTURE_COOKIE','String').SetString( 'Mon, 01 Dec 2099 12:00:00 GMT');
17797:   EXPIRED_COOKIE'.SetString( 'Mon, 01 Jan 2001 12:00:00 GMT');
17798:   'SECURE_OFF', LongInt').SetInt( 0 );
17799:   'SECURE_ON', LongInt').SetInt( 2 );
17800:   'SECURE_FILTER', LongInt').SetInt( 3 );
17801:   THandle or DWord!
17802: //  astr = ansistring;
17803: CL.AddTypeS('pastr', 'ansistring');
17804: CL.AddTypeS('TFilterFunc', 'function(const s: pastr): pastr;');
17805: uses pwinit at begin
17806: //type TFilterFunc = function(const s: astr): astr;
17807: Demo: ..\maxbox3\examples2\519_powtills.txt
17808:
17809: //CL.AddConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false );
17810: //CL.AddConstantN('CASE_IGNORE','Boolean')BoolToStr( false );
17811: Procedure pwInit();
17812: Procedure Offreadln();
17813: Function Lcase( const s : pastr ) : pastr';
17814: Function Ucase( const s : pastr ) : pastr';
17815: Function CountPostVars : longword';
17816: Function GetPostVar( const name : pastr ) : pastr';
17817: Function GetPostVarL( const name : pastr; filter : TFilterFunc ) : pastr';
17818: Function GetPostVar_S( const name : pastr; Security : integer ) : pastr';
17819: Function GetPostVar_SF( const name : pastr; Security : integer ) : pastr';
17820: Function GetPostVarAsFloat( const name : pastr ) : double';
17821: Function GetPostVarAsInt( const name : pastr ) : longint';
17822: Function GetPostVar_SafeHTML( const name : pastr ) : pastr';
17823: Function FetchPostVarName( idx : longword ) : pastr';
17824: Function FetchPostVarVal( idx : longword ) : pastr';
17825: Function FetchPostVarValL( idx : longword; filter : TFilterFunc ) : pastr';
17826: Function FetchPostVarName_S( idx : longword; Security : integer ) : pastr';
17827: Function FetchPostVarVal_S( idx : longword; Security : integer ) : pastr';
17828: Function IsPostVar( const name : pastr ) : boolean';
17829: Function CountAny : longword';
17830: Function GetAny( const name : pastr ) : pastr';
17831: Function GetAnyL( const name : pastr; filter : TFilterFunc ) : pastr';
17832: Function GetAny_S( const name : pastr; Security : integer ) : pastr';
17833: Function GetAnyAsFloat( const name : pastr ) : double';
17834: Function GetAnyAsInt( const name : pastr ) : longint';
17835: Function IsAny( const name : pastr ) : byte';
17836: Function CountCookies : longword';
17837: Function FetchCookieName( idx : longword ) : pastr';
17838: Function FetchCookieVal( idx : longword ) : pastr';
17839: Function FetchCookieValL( idx : longword; filter : TFilterFunc ) : pastr';
17840: Function GetCookie( const name : pastr ) : pastr';
17841: Function GetCookieL( const name : pastr; filter : TFilterFunc ) : pastr';
17842: Function GetCookieAsFloat( const name : pastr ) : double';
17843: Function GetCookieAsInt( const name : pastr ) : longint';
17844: Function IsCookie( const name : pastr ) : boolean';
17845: Function SetCookie( const name, value : pastr ) : boolean';
17846: Function SetCookieAsFloat( const name : pastr; value : double ) : boolean';
17847: Function SetCookieAsInt( const name : pastr; value : longint ) : boolean';
17848: Function SetCookieEx( const name, value, path, domain, expiry : pastr ) : boolean';
17849: Function SetCookieAsFloatEx( const name:pastr;value : double; const path,domain,expiry:pastr ):bool;
17850: Function SetCookieAsIntEx( const name:pastr;value : longint; const path,domain,expiry:pastr ):bool;
17851: Function UnsetCookie( const name : pastr ) : boolean';
17852: Function UnsetCookieEx( const name, path, domain : pastr ) : boolean';
17853: Function FilterHtml( const input : pastr ) : pastr';
17854: Function FilterHtml_S( const input : pastr; security : integer ) : pastr';
17855: Function TrimBadChars( const input : pastr ) : pastr';
17856: Function TrimBadFile( const input : pastr ) : pastr';
17857: Function TrimBadDir( const input : pastr ) : pastr';
17858: Function TrimBad_S( const input : pastr; security : integer ) : pastr';
17859: Function CountHeaders : longword';
17860: Function FetchHeaderName( idx : longword ) : pastr';
17861: Function FetchHeaderVal( idx : longword ) : pastr';
17862: Function GetHeader( const name : pastr ) : pastr';
17863: Function IsHeader( const name : pastr ) : boolean';
17864: Function SetHeader( const name, value : pastr ) : boolean';
17865: Function UnsetHeader( const name : pastr ) : boolean';
17866: Function PutHeader( const header : pastr ) : boolean';
17867: Procedure OutL( const s : pastr );
17868: Procedure OutLn( const s : pastr );
17869: Procedure OutA( args : array of const );
17870: Procedure OutF( const s : pastr );
17871: Procedure OutLnF( const s : pastr );
17872: Procedure OutFF( const s : pastr );
17873: Procedure OutF_FI( const s : pastr; HTMLFilter : boolean );
17874: Procedure OutLnFF( const s : pastr );

```

```

17875: Procedure OutLnF_FI( const s : pastr; HTMLFilter : boolean );
17876: Function FileOut( const fname : pastr ) : word';
17877: Function ResourceOut( const fname : pastr ) : word';
17878: Procedure BufferOut( const buff, len : LongWord );
17879: Function TemplateOut( const fname : pastr; HtmlFilter : boolean ) : word';
17880: Function TemplateOut1( const fname : pastr ) : word';
17881: Function TemplateOut2( const fname : pastr; filter : TFilterFunc ) : word';
17882: Function TemplateOut3( const fname : pastr; HtmlFilter : boolean ) : word';
17883: Function TemplateRaw( const fname : pastr ) : word';
17884: Function Fmt( const s : pastr ) : pastr );
17885: Function Fmt1( const s : pastr; filter : TFilterFunc ) : pastr );
17886: Function Fmt2( const s : pastr ) : pastr );
17887: Function Fmt_SF( const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecurity:int ):pastr;
17888: Function Fmt_SFL( const s:pastr; HTMLFilter:boolean; FilterSecurity, TrimSecurity : integer ) : pastr );
17889: Function CountRtiVars : longword );
17890: Function FetchRtiName( idx : longword ) : pastr );
17891: Function FetchRtiVal( idx : longword ) : pastr );
17892: Function GetRti( const name : pastr ) : pastr );
17893: Function GetRtiAsFloat( const name : pastr ) : double );
17894: Function GetRtiAsInt( const name : pastr ) : longint );
17895: Function IsRti( const name : pastr ) : boolean );
17896: Procedure SetRTI( const name, value : pastr );
17897: Function FetchUpfileName( idx : longword ) : pastr );
17898: Function GetUpfileName( const name : pastr ) : pastr );
17899: Function GetUpfileSize( const name : pastr ) : longint );
17900: Function GetUpFileType( const name : pastr ) : pastr );
17901: Function CountUpfiles : longword );
17902: Function IsUpfile( const name : pastr ) : boolean );
17903: Function SaveUpfile( const name, fname : pastr ) : boolean );
17904: Function CountVars : longword );
17905: Function FetchVarName( idx : longword ) : pastr );
17906: Function FetchVarVal( idx : longword ) : pastr );
17907: Function FetchVarVal1( idx : longword; filter : TFilterFunc ) : pastr );
17908: Function GetVar( const name : pastr ) : pastr );
17909: Function GetVar1( const name : pastr; filter : TFilterFunc ) : pastr );
17910: Function GetVar_S( const name : pastr; security : integer ) : pastr );
17911: Function GetVarAsFloat( const name : pastr ) : double );
17912: Function GetVarAsInt( const name : pastr ) : longint );
17913: Procedure SetVar( const name, value : pastr );
17914: Procedure SetVarAsFloat( const name : pastr; value : double );
17915: Procedure SetVarAsInt( const name : pastr; value : longint );
17916: Function IsVar( const name : pastr ) : byte );
17917: Procedure UnsetVar( const name : pastr );
17918: Function LineEndToBR( const s : pastr ) : pastr );
17919: Function RandomStr( len : longint ) : pastr );
17920: Function XorCrypt( const s : pastr; key : byte ) : pastr );
17921: Function CountCfgVars : longword );
17922: Function FetchCfgVarName( idx : longword ) : pastr );
17923: Function FetchCfgVarVal( idx : longword ) : pastr );
17924: Function IsCfgVar( const name : pastr ) : boolean );
17925: Function SetCfgVar( const name, value : pastr ) : boolean );
17926: Function GetCfgVar( const name : pastr ) : pastr );
17927: Procedure ThrowErr( const s : pastr );
17928: Procedure ThrowWarn( const s : pastr );
17929: Procedure ErrWithHeader( const s : pastr );
17930: CL.AddTypeS('TWebVar', 'record name : pastr; value : pastr; end');
17931: CL.AddTypeS('TWebVars', 'array of TWebVar');
17932: Function iUpdateWebVar(var webv : TWebVars; const name, value:pastr; upcased:boolean):boolean;
17933: Function iAddWebCfgVar( const name, value : pastr ) : boolean );
17934: Procedure iAddWebVar( var webv : TWebVars; const name, value : pastr );
17935: Procedure iSetRTI( const name, value : pastr );
17936: Function iCustomSessUnitSet : boolean );
17937: Function iCustomCfgUnitSet : boolean );
17938: end;
17939:
17940: procedure SIRегистre_W32VersionInfo(CL: TPSPascalCompiler);
17941: begin
17942:   SIRегистre_TProjectVersionInfo(CL);
17943:   Function MSLanguageToHex( const s : string ) : string );
17944:   Function MSHexToLanguage( const s : string ) : string );
17945:   Function MSCharacterSetToHex( const s : string ) : string );
17946:   Function MSHexToCharacterSet( const s : string ) : string );
17947:   Function MSLanguages : TStringList );
17948:   Function MSHexLanguages : TStringList );
17949:   Function MSCharacterSets : TStringList );
17950:   Function MSHexCharacterSets : TStringList );
17951: end;
17952:
17953: procedure SIRегистre_IpUtils(CL: TPSPascalCompiler);
17954: begin
17955:   TIpHandle', 'Cardinal');
17956:   TIpMD5StateArray', 'array[0..3] of DWORD');
17957:   CL.AddTypeS('TIpMD5CountArray', 'array[0..1] of DWORD');
17958:   TIpMD5ByteBuf', 'array[0..63] of Byte');
17959:   TIpMD5LongBuf', 'array[0..15] of DWORD');
17960:   TIpMD5Digest', 'array[0..15] of Byte');
17961:   TIpLineTerminator', '( ltNone, ltCR, ltLF, ltCRLF, ltOther )');
17962:   TIpMD5Context', 'record State : TIpMD5StateArray; Count : TIpMD5';
17963:     +'CountArray; ByteBuf : TIpMD5ByteBuf; end');

```

```

17964: CL.AddClassN(CL.FindClass('TOBJECT'), 'EIpBaseException');
17965: CL.AddClassN(CL.FindClass('TOBJECT'), 'EIpAccessException');
17966: CL.AddClassN(CL.FindClass('TOBJECT'), 'EIpHtmlException');
17967: SIRegister_TIpBaseAccess(CL);
17968: SIRegister_TIpBasePersistent(CL);
17969: //TIPComponentClass', 'class of TIpBaseComponent');
17970: SIRegister_TIpBaseComponent(CL);
17971: SIRegister_TIpBaseWinControl(CL);
17972: Function InClassA( Addr : LongInt ) : Boolean';
17973: Function InClassB( Addr : LongInt ) : Boolean';
17974: Function InClassC( Addr : LongInt ) : Boolean';
17975: Function InClassD( Addr : LongInt ) : Boolean';
17976: Function InMulticast( Addr : LongInt ) : Boolean';
17977: Function IpCharCount( const Buffer, BufSize : DWORD; C : AnsiChar ) : DWORD';
17978: Function IpCompStruct( const S1, S2, Size : Cardinal ) : Integer';
17979: Function IpMaxInt( A, B : Integer ) : Integer';
17980: Function IpMinInt( A, B : Integer ) : Integer';
17981: Procedure IpSafeFree( var Obj: TObject );
17982: Function IpShortVersion : string';
17983: Function IpInternetSumPrim( var Data, DataSize, CurCrc : DWORD ) : DWORD';
17984: Function IpInternetSumOfStream( Stream : TStream; CurCrc : DWORD ) : DWORD';
17985: Function IpInternetSumOfFile( const FileName : string ) : DWORD';
17986: Function MD5SumOfFile( const FileName : string ) : string';
17987: Function MD5SumOfStream( Stream : TStream ) : string';
17988: Function MD5SumOfStreamDigest( Stream : TStream ) : TIpMD5Digest';
17989: Function MD5SumOfString( const S : string ) : string';
17990: Function MD5SumOfStringDigest( const S : string ) : TIpMD5Digest';
17991: Function SafeYield : LongInt';
17992: Function AllTrimSpaces( Strng : string ) : string';
17993: Function IpCharPos( C : AnsiChar; const S : string ) : Integer';
17994: Function CharPosIdx( C : AnsiChar; const S : string; Idx : Integer ) : Integer';
17995: Function NthCharPos( C : AnsiChar; const S : string; Nth : Integer ) : Integer';
17996: Function RCharPos( C : AnsiChar; const S : string ) : Integer';
17997: Function RCharPosIdx( C : AnsiChar; const S : string; Idx : Integer ) : Integer';
17998: Function RNthCharPos( C : AnsiChar; const S : string; Nth : Integer ) : Integer';
17999: Function IpRPos( const Substr : string; const S : string ) : Integer';
18000: Function IpPosIdx( const Substr, S : string; Idx : Integer ) : Integer';
18001: ACharSet', 'set of AnsiChar');
18002: TipAddrRec', 'record Scheme : string; UserName : string; Password string; Authority : string;
18003: Port : string; Path : string; Fragment : string; Query : string; QueryDelim : AnsiChar; end');
18004: Procedure Initialize( var AddrRec : TipAddrRec );
18005: Procedure Finalize( var AddrRec : TipAddrRec );
18006: Function ExtractEntityName( const NamePath : string ) : string';
18007: Function ExtractEntityPath( const NamePath : string ) : string';
18008: Function IpParseURL( const URL : string; var Rslt : TipAddrRec ) : Boolean';
18009: Function BuildURL( const OldURL, NewURL : string ) : string';
18010: Function PutEscapes( const S : string; EscapeSet : ACharSet ) : string';
18011: Function RemoveEscapes( const S : string; EscapeSet : ACharSet ) : string';
18012: Procedure SplitParams( const Params : string; Dest : TStrings );
18013: Function NetToDOSPath( const PathStr : string ) : string';
18014: Function DOSToNetPath( const PathStr : string ) : string';
18015: Procedure SplitHttpResponse( const S : string; var V, MsgID, Msg : string );
18016: Procedure FieldFix( Fields : TStrings );
18017: Function AppendSlash( APath : string ) : string';
18018: Function RemoveSlash( APath : string ) : string';
18019: Function GetParentPath( const Path : string ) : string';
18020: Function GetLocalContent( const TheFileName : string ) : string';
18021: Function IPDirExists( Dir : string ) : Boolean';
18022: Function GetTemporaryFile( const Path : string ) : string';
18023: Function GetTemporaryPath : string';
18024: Function AppendBackSlash( APath : string ) : string';
18025: Function IpRemoveBackSlash( APath : string ) : string';
18026: Function INetDateToStrToDate( const DateStr : string ) : TDateTime';
18027: Function DateToString( Date : TDateTime ) : string';
18028: Function IpTimeZoneBias : Integer';
18029: Procedure SplitCookieFields( const Data : string; Fields : TStrings );
18030: end;
18031:
18032: procedure SIRegister_LrtPoTools(CL: TPSPascalCompiler);
18033: begin
18034:   CL.AddTypeS('TPOStyle', '( postStandard, postPropName, postFull )');
18035:   Procedure Lrt2Po( const LRTfile : string; PostStyle : TPOStyle );
18036:   Procedure CombinePoFiles( SL : TStrings; const FName : string );
18037: end;
18038:
18039: procedure SIRegister_GPS(CL: TPSPascalCompiler);
18040: begin
18041:   CL.AddConstantN('MAX_SATS', 'LongInt').SetInt( 12 );
18042:   GPSMSG_START', 'String').SetString( '$' );
18043:   GPSMSG_STOP', 'String').SetString( '*' );
18044:   SEC_BETWEEN_SEG', 'LongInt').SetInt( 5 );
18045:   CL.AddTypeS('TSatellite', 'record Identification : Shortint; Elevation : Shor
18046:     +tint; Azimut : Smallint; SignLevel : Smallint; end');
18047:   CL.AddTypeS('TSatellites', 'array[1..12] of TSatellite');
18048:   //TSatellites = array[1..MAX_SATS] of TSatellite;
18049:   TGPSSatEvent', 'Procedure ( Sender : TObject; NbSat, NbSatUse: Shortint; Sats : TSatellites )');
18050:   TGPSDatas', 'record Latitude : Double; Longitude : Double; Heigh
18051:     tAboveSea : Double; Speed : Double; UTCTime : TDateTime; Valid : Boolean;
18052:     NbrSats : Shortint; NbrSatsUsed : Shortint; Course : Double; end');

```

```

18053: CL.AddTypeS('TGPSDatasEvent', 'Procedure ( Sender : TObject; GPSDatas : TGPSDatas)');
18054: CL.AddTypeS('TMsgGP', '( msgGP, msgGPGGA, msgGPGLL, msgGPGSV, msgGPRMA, msgGPRMC, msgGPZDA )');
18055: CL.AddTypeS('TSpeedUnit', '( suKilometre, suMile, suNauticalMile )');
18056: SIRegister_TGPSLink(CL);
18057: SIRegister_TCustomGPS(CL);
18058: SIRegister_TGPS(CL);
18059: SIRegister_TGPSSoGPX(CL);
18060: SIRegister_TGPSSpeed(CL);
18061: SIRegister_TGPSSatellitesPosition(CL);
18062: SIRegister_TGPSSatellitesReception(CL);
18063: SIRegister_TGPSCompass(CL);
18064: //Procedure Register( );
18065: Function IndexMsgGP( StrMsgGP : String ) : TMsgGP';
18066: Function StrCoordToAngle( Point : Char; Angle : String ) : Double';
18067: Function StrTimeToTime( const Time : String ) : TDateTime';
18068: Function StrToInteger( const Str : String ) : Integer';
18069: Function StrToReal( const Str : String ) : Extended';
18070: Function GPSRotatePoint( Angle : Double; Ct, Pt : TPoint ) : TPoint';
18071: Procedure LoadRessource( RessourceName : String; ImageList : TImageList );
18072: end;
18073:
18074: procedure SIRegister_NMEA(CL: TPSPascalCompiler);
18075: begin
18076: NMEADataArray', 'array of string');
18077: Procedure TrimNMEA( var S : string );
18078: Procedure ExpandNMEA( var S : string );
18079: Function ParseNMEA( S : string ) : NMEADataArray';
18080: Function ChkValidNMEA( S : string ) : Boolean';
18081: Function IdNMEA( S : string ) : string';
18082: Function ChkSumNMEA( const S : string ) : string';
18083: Function PosInDeg( const PosStr : string ) : Double';
18084: Function DateNMEA( const StrD, StrT : string ) : TDateTime';
18085: Function SysClockSet( const StrD, StrT : string ) : Boolean';
18086: function Ticks2Secs(Ticks : LongInt) : LongInt';
18087: function Secs2Ticks(Secs : LongInt) : LongInt';
18088: function MSecs2Ticks(MSecs : LongInt) : LongInt';
18089: end;
18090:
18091: procedure SIRegister_SortUtils(CL: TPSPascalCompiler);
18092: begin
18093: CL.AddTypeS('SortType1', 'Byte');
18094: CL.AddTypeS('SortType2', 'Double');
18095: CL.AddTypeS('SortType3', 'DWord');
18096: //CL.AddTypeS('PDWordArray', '^DWordArray // will not work');
18097: CL.AddTypeS('TDataRecord4', 'record Value : Integer; Data : Integer; end');
18098: Function('Procedure QuickSort( var List : array of SortType1; Min, Max : Integer)');
18099: Procedure QuickSortDWord( var List : array of SortType3; Min, Max : Integer );
18100: Procedure QuickSortDataRecord4( var List : array of TDataRecord4; Count : Integer );
18101: Procedure HeapSort( var List : array of SortType1; Count : DWord; FirstNeeded : DWord );
18102: Function QuickSelect( var List : array of SortType1; Min, Max, Wanted : Integer ) : SortType1';
18103: Function QuickSelectDouble( var List : array of SortType2; Min, Max, Wanted : Integer ) : SortType2';
18104: Function QuickSelectDWord( var List : array of SortType3; Min, Max, Wanted : Integer ) : SortType3';
18105: end;
18106:
18107: procedure SIRegister_BitmapConversion(CL: TPSPascalCompiler);
18108: begin
18109: // TMatrix3x3 = array[1..3,1..3] of Double;
18110: // TMatrix4x4 = array[1..4,1..4] of Double;
18111: CL.AddTypeS('TMatrix3x31', 'array[1..3] of Double');
18112: CL.AddTypeS('TMatrix3x3', 'array[1..3] of TMatrix3x31');
18113: CL.AddTypeS('TMatrix4x41', 'array[1..4] of Double');
18114: CL.AddTypeS('TMatrix4x4', 'array[1..4] of TMatrix4x41');
18115: Procedure ColorTransform( A, B, C : Byte; out X, Y, Z : Byte; const T : TMatrix4x4 );
18116: Procedure ColorTransform1( A, B, C : Byte; out X, Y, Z : Float; const T : TMatrix4x4 );
18117: Procedure ColorTransform2( const A, B, C : Float; out X, Y, Z : Byte; const T : TMatrix4x4 );
18118: Procedure ColorTransformHSI2RGB( H, S, I : Byte; out R, G, B : Byte );
18119: Procedure ColorTransformRGB2HSI( R, G, B : Byte; out H, S, I : Byte );
18120: Procedure ColorTransformRGB2Lab( R, G, B : Byte; out L, a_, b_ : Byte );
18121: Procedure ColorTransformLab2RGB( L, a_, b_ : Byte; out R, G, B : Byte );
18122: Procedure ColorTransformRGB2LOCO( R, G, B : Byte; out S0, S1, S2 : Byte );
18123: Procedure ColorTransformLOCO2RGB( S0, S1, S2 : Byte; out R, G, B : Byte );
18124: Procedure ConvertColorSpace( Image : TLinarBitmap; const T : TMatrix4x4; NewImage : TLinarBitmap );
18125: //Procedure
18126: ConvertColorSpace1(Image:TLinarBitmap;ColorTransform:TColorTransformProc;NewImage:TLinarBitmap);
18127: Procedure ConvertToGrayscale( const Image, GrayImage : TLinarBitmap );
18128: Procedure ConvertToGrayscale1( const Image : TLinarBitmap );
18129: end;
18130: procedure SIRegister_ZDbcUtils(CL: TPSPascalCompiler);
18131: begin
18132: { TZSQLType = (zsqlstUnknown, zsqlstBoolean, zsqlstByte, zsqlstShort, zsqlstInteger, zsqlstLong,
18133: zsqlstFloat, zsqlstDouble, zsqlstBigDecimal, zsqlstString, zsqlstUnicodeString, zsqlstBytes,
18134: zsqlstDate, zsqlstTime, zsqlstTimestamp, zsqlstDataSet, zsqlstGUID,
18135: stAsciiStream, stUnicodeStream, stBinaryStream); }
18136: Function ResolveConnectionProtocol( Url : string; SupportedProtocols : TStringDynArray ) : string';
18137: //Procedure ResolveDatabaseUrl( const Url: string; Info:TStrings; var HostName:string; var
18138: Port:Integer;var Database : string; var UserName : string; var Password : string; ResultInfo : TStrings );
18139: Function CheckConversion( InitialType : TZSQLType; ResultType : TZSQLType ) : Boolean';
18139: Function DefineColumnType( ColumnType : TZSQLType ) : string';

```

```

18140: Procedure RaiseSQLEception( E : Exception)'');
18141: //Procedure CopyColumnInfo( FromList : TObjectList; ToList : TObjectList)');
18142: Function ToLikeString( const Value : string) : string');
18143: Function GetSQLHexWideString( Value : PChar; Len : Integer; ODBC : Boolean) : WideString');
18144: Function GetSQLHexAnsiString( Value : PChar; Len : Integer; ODBC : Boolean) : RawByteString');
18145: Function GetSQLHexString( Value : PChar; Len : Integer; ODBC : Boolean) : String');
18146: Function WideStringStream( const AString : WideString) : TStream');
18147: function ConvertAdoToTypeName(FieldType: SmallInt): string;
18148: function GetTableName(const AField: TField): string;');
18149: function GetFieldName(const AField: TField): string;');
18150: end;
18151:
18152: procedure SIRegister_JclTD32(CL: TPSPascalCompiler);
18153: CL.AddTypeS('TSymbolInfo', 'record Size : Word; SymbolType : Word; end');
18154: //CL.AddTypeS('PSymbolInfos', '^TSymbolInfos // will not work');
18155: SIRegister_TJclModuleInfo(CL);
18156: SIRegister_TJclLineInfo(CL);
18157: SIRegister_TJclSourceModuleInfo(CL);
18158: SIRegister_TJclSymbolInfo(CL);
18159: SIRegister_TJclProcSymbolInfo(CL);
18160: CL.AddClassN(CL.FindClass('TOBJECT'), 'TJclLocalProcSymbolInfo');
18161: CL.AddClassN(CL.FindClass('TOBJECT'), 'TJclGlobalProcSymbolInfo');
18162: SIRegister_TJclTD32InfoParser(CL);
18163: SIRegister_TJclTD32InfoScanner(CL);
18164: SIRegister_TJclPeBorTD32Image(CL);
18165: end;
18166:
18167: procedure SIRegister_JvIni(CL: TPSPascalCompiler);
18168: begin
18169: CL.AddTypeS('TReadObjectEvent', 'Function ( Sender : TObject; const Section,
18170: +Item, Value : string) : TObject');
18171: TWriteObjectEvent', 'Procedure ( Sender : TObject; const Section, Item: string; Obj: TObject)');
18172: Function StringToFontStyles( const Styles : string) : TFontStyles');
18173: Function FontStylesToString( Styles : TFontStyles) : string');
18174: Function FontToString( Font : TFont) : string');
18175: Procedure StringToFont( const Str : string; Font : TFont)');
18176: Function RectToStr( Rect : TRect) : string');
18177: Function StrToRect( const Str : string; const Def : TRect) : TRect');
18178: Function JPointToStr( P : TPoint) : string');
18179: Function JStrToPoint( const Str : string; const Def : TPoint) : TPoint');
18180: Function DefProfileName : string');
18181: Function DefLocalProfileName : string');
18182: CL.AddConstantN('idnListItem','String').SetString( 'Item');
18183: end;
18184:
18185: procedure SIRegister_JvHtControls(CL: TPSPascalCompiler);
18186: begin
18187: Procedure ItemHtDrawEx( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text :
18188: string; const HideSelColor : Boolean; var PlainItem : string; var Width : Integer; CalcWidth : Boolean)');
18189: Function ItemHtDraw( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text : string;
18190: const HideSelColor : Boolean) : string');
18191: Function ItemHtWidth( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text : string;
18192: const HideSelColor : Boolean) : Integer');
18193: Function ItemHtPlain( const Text : string) : string');
18194: Procedure ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:string);
18195: Function IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:string;MouseX,MouseY:Integer;var
18196: HyperLink:string):Bool;
18197: end;
18198: procedure SIRegister_NeuralNetwork(CL: TPSPascalCompiler);
18199: begin
18200: CL.AddClassN(CL.FindClass('TOBJECT'), 'TNeuron');
18201: CL.AddTypeS('TSynapse', 'record W : Real; Connection : TNeuron; end');
18202: CL.AddTypeS('TAcson', 'record Alfa : Real; Beta : Real; Gama : Real; end');
18203: SIRegister_TNeuron(CL);
18204: SIRegister_TNeuronLayer(CL);
18205: SIRegister_TNeuralNet(CL);
18206: end;
18207: procedure SIRegister_StExpr(CL: TPSPascalCompiler);
18208: begin
18209: //CL.AddTypeS('PStFloat', '^TStFloat // will not work');
18210: TStMethod0Param', 'Function : TStFloat');
18211: TStMethod1Param', 'Function ( Value1 : TStFloat) : TStFloat');
18212: TStMethod2Param', 'Function ( Value1, Value2 : TStFloat) : TStFloat');
18213: TStMethod3Param', 'Function ( Value1, Value2, Value3 : TStFloat) : TStFloat');
18214: TStGetIdentValueEvent', 'Procedure ( Sender : TObject; const Ide'
18215: +'ntifier : AnsiString; var Value : TStFloat)');
18216: TStToken', '( ssStart, ssInIdent, ssInNum, ssInSign, ssInExp, ss
18217: +'Eol, ssNum, ssIdent, ssLPar, ssRPar, ssComma, ssPlus, ssMinus, ssTimes, ssDiv, ssEqual, ssPower )');
18218: SIRegister_TStExpression(CL);
18219: TStExprErrorEvent', 'Procedure ( Sender : TObject; ErrorNumber : '
18220: +' LongInt; const ErrorStr : AnsiString)');
18221: Function AnalyzeExpr( const Expr : AnsiString) : Double';
18222: Procedure TpVal( const S : AnsiString; var V : Extended; var Code : Integer)');
18223: end;
18224: procedure SIRegister_GR32_Containers(CL: TPSPascalCompiler);

```

```

18225: begin
18226:   CL.AddConstantN('BUCKET_MASK','LongWord').SetUInt( $FF);
18227:   CL.AddConstantN('BUCKET_COUNT','LongInt').SetInt( BUCKET_MASK + 1);
18228:   Procedure SmartAssign( Src, Dst : TPersistent; TypeKinds : TTypeKinds );
18229:   Procedure Advance( var Node : TLinkedNode; Steps : Integer );
18230: end;
18231:
18232: procedure SIRегистre_StSaturn(CL: TPPascalCompiler);
18233: begin
18234:   TStJupSatPos', 'record X: double; Y: Double; end');
18235:   TStJupSats', 'record Io: TStJupSatPos;
18236:     Europa:TStJupSatPos;Ganymede:TStJupSatPos;Callisto:TStJupSatPos;end';
18237:     Function ComputeSaturn( JD : Double ) : TStEclipticalCord';
18238:     Function ComputePluto( JD : Double ) : TStEclipticalCord';
18239:     Function ComputeVenus( JD : Double ) : TStEclipticalCord';
18240:     Function ComputeMars( JD : Double ) : TStEclipticalCord';
18241:     Function ComputeMercury( JD : Double ) : TStEclipticalCord';
18242:     Function ComputeJupiter( JD : Double ) : TStEclipticalCord';
18243:     Function ComputeUranus( JD : Double ) : TStEclipticalCord';
18244:     Function ComputeNeptune( JD : Double ) : TStEclipticalCord';
18245:   function GetJupSats(JD : TDateTime; HighPrecision, Shadows : Boolean) : TStJupSats;');
18246: end;
18247: procedure SIRегистre_JclParseUses(CL: TPPascalCompiler);
18248: begin
18249:   CL.AddClassN(CL.FindClass('TOBJECT'),'EUsesListError');
18250:   Function CreateGoal( Text : PChar ) : TCustomGoal;
18251: end;
18252:
18253: procedure SIRегистre_JvFinalize(CL: TPPascalCompiler);
18254: begin
18255: //type
18256: //  TFinalizeProc = procedure;
18257:   CL.AddTypeS('TFinalizeProc', 'procedure');
18258:   Procedure AddFinalizeProc( const UnitName : string; FinalizeProc : TFinalizeProc );
18259:   Function AddFinalizeObject( const UnitName : string; Instance : TObject ) : TObject;
18260:   Function AddFinalizeObjectNil( const UnitName : string; var Reference: TObject ) : TObject;
18261:   Function AddFinalizeFreeAndNil( const UnitName : string; var Reference: TObject ) : TObject;
18262:   Function AddFinalizeMemory( const UnitName : string; Ptr : __Pointer ) : __Pointer;
18263:   Function AddFinalizeMemoryNil( const UnitName : string; var Ptr: __Pointer ) : __Pointer;
18264:   Procedure FinalizeUnit( const UnitName : string );
18265: end;
18266:
18267: procedure SIRегистre_BigIni(CL: TPPascalCompiler);
18268: begin
18269:   CL.AddConstantN('InitextBufferSize','LongWord').SetUInt( $7000);
18270:   CL.AddConstantN('ciniCount','String').SetString( 'Count' );
18271:   TEraseSectionCallback', 'Function ( const sectionName : string; '
18272:   const sl1, sl2 : TStringList ) : Boolean';
18273:   SIRегистre_TCommaSeparatedInfo(CL);
18274:   SIRегистre_TSectionList(CL);
18275:   SIRегистre_TBigIniFile(CL);
18276:   SIRегистre_TBiggerIniFile(CL);
18277:   SIRегистre_TAppIniFile(CL);
18278:   SIRегистre_TLibIniFile(CL);
18279:   Function ModuleName( getLibraryName : Boolean ) : String';
18280: end;
18281:
18282: procedure SIRегистre_ShellCtrls(CL: TPPascalCompiler);
18283: begin
18284:   CL.AddTypeS('TRoot', 'string');
18285:   CL.AddTypeS('TRootFolder', '( rfDesktop, rfMyComputer, rfNetwork, rfRecycleBin'
18286:   +'n, rfAppData, rfCommonDesktopDirectory, rfCommonPrograms, rfCommonStartMen'
18287:   +'u, rfCommonStartup, rfControlPanel, rfDesktopDirectory, rfFavorites, rfFon'
18288:   +'ts, rfInternet, rfPersonal, rfPrinters, rfPrintHood, rfPrograms, rfRecent, '
18289:   +' rfSendTo, rfStartMenu, rfStartup, rfTemplates )');
18290:   TShellFolderCapability', '( fcCanCopy, fcCanDelete, fcCanLink, f'
18291:   +'cCanMove, fcCanRename, fcDropTarget, fcHasPropSheet )';
18292:   TShellFolderCapabilities', 'set of TShellFolderCapability';
18293:   TShellFolderProperty', '( fpCut, fpIsLink, fpReadOnly, fpShared, '
18294:   +' fpFileSystem, fpFileSystemAncestor, fpRemovable, fpValidate )';
18295:   TShellFolderProperties', 'set of TShellFolderProperty';
18296:   TShellObjectType', '( otFolders, otNonFolders, otHidden )';
18297:   TShellObjectTypes', 'set of TShellObjectType';
18298:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInvalidPath');
18299:   SIRегистre_IShellCommandVerb(CL);
18300:   SIRегистre_TShellFolder(CL);
18301:   TNotifyFilter', '( nfFileNameChange, nfDirNameChange, nfAttribut'
18302:   +'eChange, nfSizeChange, nfWriteChange, nfSecurityChange )';
18303:   TNotifyFilters', 'set of TNotifyFilter';
18304:   SIRегистre_TShellChangeThread(CL);
18305:   SIRегистre_TCustomeShellChangeNotifier(CL);
18306:   SIRегистre_TShellChangeNotifier(CL);
18307:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomShellComboBox');
18308:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomShellListView');
18309:   TAddFolderEvent', 'Procedure ( Sender : TObject; AFolder : TShel'
18310:   +'lFolder; var CanAdd : Boolean )';
18311:   TGetImageIndexEvent', 'Procedure ( Sender : TObject; Index : Int'
18312:   +'eger; var ImageIndex : Integer )';

```

```

18313:  SIRегистер_TCustomShellTreeView(CL);
18314:  SIRегистер_TShellTreeView(CL);
18315:  SIRегистер_TCustomShellComboBox(CL);
18316:  SIRегистер_TShellComboBox(CL);
18317:  SIRегистер_TCustomShellListView(CL);
18318:  SIRегистер_TShellListView(CL);
18319:  Procedure InvokeContextMenu( Owner : TWinControl; AFolder : TShellFolder; X, Y : Integer)'');
18320: end;
18321:
18322: procedure SIRегистер_fmath(CL: TPSpascalCompiler);
18323: begin
18324:   CL.AddTypeS('Float', 'Double');
18325:   'FN_OK', 'LongInt').SetInt( 0 );
18326:   CL.AddConstantN('FN_DOMAIN', 'LongInt').SetInt( - 1 );
18327:   'FN_SING', 'LongInt').SetInt( - 2 );
18328:   'FN_OVERFLOW', 'LongInt').SetInt( - 3 );
18329:   'FN_UNDERFLOW', 'LongInt').SetInt( - 4 );
18330:   'FN_TLOSS', 'LongInt').SetInt( - 5 );
18331:   'FN_PLOSS', 'LongInt').SetInt( - 6 );
18332: //CL.AddConstantN('NFACT', 'LongInt').SetInt( 33 );
18333: Function MathError : Integer';
18334: Function FMin( X, Y : Float ) : Float';
18335: Function FMax( X, Y : Float ) : Float';
18336: Function IMin( X, Y : Integer ) : Integer';
18337: Function IMax( X, Y : Integer ) : Integer';
18338: Function FSgn( X : Float ) : Integer';
18339: Function Sgn0( X : Float ) : Integer';
18340: Function DSgn( A, B : Float ) : Float';
18341: Procedure FSwap( var X, Y : Float );
18342: Procedure ISwap( var X, Y : Integer );
18343: Function fExpo( X : Float ) : Float';
18344: Function fExp2( X : Float ) : Float';
18345: Function fExp10( X : Float ) : Float';
18346: Function fLog( X : Float ) : Float';
18347: Function fLog2( X : Float ) : Float';
18348: Function fLog10( X : Float ) : Float';
18349: Function fLogA( X, A : Float ) : Float';
18350: Function fIntPower( X : Float; N : Integer ) : Float';
18351: Function fPower( X, Y : Float ) : Float';
18352: Function Pythag( X, Y : Float ) : Float';
18353: Function FixAngle( Theta : Float ) : Float';
18354: Function fTan( X : Float ) : Float';
18355: Function fArcSin( X : Float ) : Float';
18356: Function fArcCos( X : Float ) : Float';
18357: Function fArcTan2( Y, X : Float ) : Float';
18358: Procedure fSinCos( X : Float; var SinX, CosX : Float );
18359: Function fSinh( X : Float ) : Float';
18360: Function fCosh( X : Float ) : Float';
18361: Function fTanh( X : Float ) : Float';
18362: Function fArcSinh( X : Float ) : Float';
18363: Function fArcCosh( X : Float ) : Float';
18364: Function fArcTanh( X : Float ) : Float';
18365: Procedure fSinhCosh( X : Float; var SinhX, CoshX : Float );
18366: Function fFact( N : Integer ) : Float';
18367: Function fBinomial( N, K : Integer ) : Float';
18368: Function fGamma( X : Float ) : Float';
18369: Function fSgnGamma( X : Float ) : Integer';
18370: Function LnGamma( X : Float ) : Float';
18371: Function fIGamma( A, X : Float ) : Float';
18372: Function fJGamma( A, X : Float ) : Float';
18373: Function fBeta( X, Y : Float ) : Float';
18374: Function fIBeta( A, B, X : Float ) : Float';
18375: Function fErf( X : Float ) : Float';
18376: Function fErfc( X : Float ) : Float';
18377: Function fPBinom(N: Integer; P: Float; K : Integer) : Float';
18378: Function FBinom(N: Integer; P: Float; K : Integer) : Float';
18379: Function PPoisson( Mu : Float; K : Integer ) : Float';
18380: Function FPoisson( Mu : Float; K : Integer ) : Float';
18381: Function fDNorm( X : Float ) : Float';
18382: Function FNorm( X : Float ) : Float';
18383: Function PNorm( X : Float ) : Float';
18384: Function InvNorm( P : Float ) : Float';
18385: Function fDStudent( Nu : Integer; X : Float ) : Float';
18386: Function FStudent( Nu : Integer; X : Float ) : Float';
18387: Function PStudent( Nu : Integer; X : Float ) : Float';
18388: Function fDKhi2( Nu : Integer; X : Float ) : Float';
18389: Function FKhi2( Nu : Integer; X : Float ) : Float';
18390: Function PKhi2( Nu : Integer; X : Float ) : Float';
18391: Function fDSnedecor( Nul, Nu2 : Integer; X : Float ) : Float';
18392: Function FSnedecor( Nul, Nu2 : Integer; X : Float ) : Float';
18393: Function PSnedecor( Nul, Nu2 : Integer; X : Float ) : Float';
18394: Function fDExp( A, X : Float ) : Float';
18395: Function FExpo( A, X : Float ) : Float';
18396: Function fDBeta( A, B, X : Float ) : Float';
18397: Function FBeta( A, B, X : Float ) : Float';
18398: Function fDGamma( A, B, X : Float ) : Float';
18399: Function FGamma( A, B, X : Float ) : Float';
18400: Procedure RMarIn( Seed1, Seed2 : Integer );
18401: Function IRanMar : LongInt );

```

```

18402: Function RanMar : Float');
18403: Function RanGaussStd : Float');
18404: Function RanGauss( Mu, Sigma : Float) : Float');
18405: end;
18406:
18407: procedure SIRegister_fcomp(CL: TPSPascalCompiler);
18408: begin
18409:   CL.AddTypeS('ComplexForm', '( Rec, Pol )');
18410:   CL.AddTypeS('TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R :'
18411:     +' Float; Theta : Float; end');
18412:   Procedure CSet( var Z : TComplex; A, B : Float; F : ComplexForm)');
18413:   Procedure CConvert( var Z : TComplex; F : ComplexForm)');
18414:   Procedure CSwap( var X, Y : TComplex)');
18415:   Function CReal( Z : TComplex) : Float');
18416:   Function CImag( Z : TComplex) : Float');
18417:   Function CAbs( Z : TComplex) : Float');
18418:   Function CArg( Z : TComplex) : Float');
18419:   Function CSgn( Z : TComplex) : Integer');
18420:   Procedure CNeg( A : TComplex; var Z : TComplex)');
18421:   Procedure CConj( A : TComplex; var Z : TComplex)');
18422:   Procedure CAdd( A, B : TComplex; var Z : TComplex)');
18423:   Procedure CSub( A, B : TComplex; var Z : TComplex)');
18424:   Procedure CDiv( A, B : TComplex; var Z : TComplex)');
18425:   Procedure CMult( A, B : TComplex; var Z : TComplex)');
18426:   Procedure CLn( A : TComplex; var Z : TComplex)');
18427:   Procedure CExp( A : TComplex; var Z : TComplex)');
18428:   Procedure CPower( A, B : TComplex; var Z : TComplex)');
18429:   Procedure CIntPower( A : TComplex; N : Integer; var Z : TComplex)');
18430:   Procedure CRealPower( A : TComplex; X : Float; var Z : TComplex)');
18431:   Procedure CSqrt( A : TComplex; var Z : TComplex)');
18432:   Procedure CRoot( A : TComplex; K, N : Integer; var Z : TComplex)');
18433:   Procedure CCSin( A : TComplex; var Z : TComplex)');
18434:   Procedure CCos( A : TComplex; var Z : TComplex)');
18435:   Procedure CTan( A : TComplex; var Z : TComplex)');
18436:   Procedure CArcSin( A : TComplex; var Z : TComplex)');
18437:   Procedure CArcCos( A : TComplex; var Z : TComplex)');
18438:   Procedure CArcTan( A : TComplex; var Z : TComplex)');
18439:   Procedure CSinh( A : TComplex; var Z : TComplex)');
18440:   Procedure CCosh( A : TComplex; var Z : TComplex)');
18441:   Procedure CTanh( A : TComplex; var Z : TComplex)');
18442:   Procedure CArcSinh( A : TComplex; var Z : TComplex)');
18443:   Procedure CArcCosh( A : TComplex; var Z : TComplex)');
18444:   Procedure CArcTanh( A : TComplex; var Z : TComplex)');
18445:   Procedure CLnGamma( A : TComplex; var Z : TComplex)');
18446: end;
18447:
18448: procedure SIRegister_XSBuiltIns(CL: TPSPascalCompiler);
18449: begin
18450:   CL.AddConstantN('SHexMarker','String').SetString( '$');
18451:   Function DateTimeToXMLTime( Value : TDateTime; ApplyLocalBias : Boolean) : WideString';
18452:   Function XMLTimeToDate( const XMLDate : WideString; AsUTCTime : Boolean): TDateTime';
18453:   Function DateTimeToXSDateTime(const Value: TDateTime; ApplyLocalBias: Boolean): TXSDateTime';
18454:   Function GetDataFromFile( AFileName : string) : string');
18455:   Function SoapFloatToStr( Value : double) : string');
18456:   Function SoapStrToFloat( Value : string) : double');
18457:   Procedure InitXSTypes');
18458: end;
18459:
18460: procedure SIRegister_CompFileIo(CL: TPSPascalCompiler);
18461: begin
18462:   SIRegister_TComponentStream(CL);
18463:   CL.AddTypeS('TComponentStream', 'TMemoryStream');
18464:   CL.AddTypeS('TComponentFormat', '( cffText, cffBinary )');
18465:   CL.AddTypeS('TComponentArray', 'array of TComponent');
18466:   TResourceNaming', rnClassNameTag, rnClassName, rnClassTag, rnNameTag, rnClass, rnName, rnTag );
18467:   CL.AddDelphiFunction('Function VOID_COMP : __Pointer');
18468:   CL.AddDelphiFunction('Function VOID_FORM : __Pointer');
18469:   CL.AddDelphiFunction('Function CreateIoForm( AForm : TCustomForm; ClassType : TFormClass) : TCustomForm');
18470:   Function GetComponentTree(Component:TComponent;pComponents:TComponentArray; bAddSelf:Boolean):LongInt';
18471:   Function pPosEx( SearchStr : PChar; Str : PChar; var Pos : LongInt) : PChar');
18472:   Function pGetTextBetween(pBuff: PChar;bSearchCode:string; eSearchCode:string;Container:TStrings) :
18473:     LongInt';
18474:   Function pComponentToString( Component : TComponent) : string');
18475:   //Function StringToComponent( Value : string; ComponentClass : TComponentClass) : TComponent');
18476:   Function StringToObjectBinaryStream( Value : string; BinStream : TMemoryStream; ResName : string):
18477:     Boolean;
18478:   Function ObjectBinaryStreamToString(BinStream:TMemoryStream;var sResult:string;
18479:     bResource:Boolean);
18480:   Function ObjectTextToBinaryStream( StrStream, BinStream : TComponentStream; ObjCount : LongInt) :
18481:     LongInt';
18482:   Function ObjectBinaryStreamToObjectTextStream( BinStream : TMemoryStream; StrStream : TMemoryStream;
18483:     bResource : Boolean) : Boolean';
18484:   Function GetResHeaderInfo( Stream : TMemoryStream; sl : TStrings) : Boolean');
18485:   Function pGetResourceName( Component : TComponent; NamingMethod : TResourceNaming) : string');
18486:   Function GetFormResourceStream(Form:TCustomForm;ResName:string; var ResourceStream:TMemoryStream):Boolean;
18487:   Function WriteComponentsToFile( FormsAndComponents : array of TComponent; FileName : string; Format :
18488:     TComponentFormat; StoreComponentNames : Boolean) : Boolean');
18489:   Function ReadComponentsFromFile( FormsAndComponents : array of TComponent; FileName: string): Boolean;');
18490:   //Function ReadComponentsFromFile( FileName : string; pComponents : array of TPComponent;
18491:     ComponentClasses : array of TComponentClass) : Boolean;');

```

```

18485: Function ReadComponentTreeFromFile( FormOrComponent : TComponent; FileName : string ) : Boolean';
18486: Function WriteComponentToFile( Component : TComponent; FileName : string; Format : TComponentFileFormat;
  StoreComponentName : Boolean ) : Boolean';
18487: Function WriteComponentTreeToFile( FormOrComponent : TComponent; FileName : string; Format :
  TComponentFileFormat; StoreComponentName : Boolean ) : Boolean';
18488: Function ReadComponentFromFile4( FormOrComponent : TComponent; FileName : string ) : Boolean';
18489: Function ReadFormFromFile( pInstance : TComponent; FormClass : TFormClass; FileName : string ) : Boolean';
18490: Function ReadComponentResourceFile5( Instance : TObject; FileName : string ) : Boolean';
18491: Function
  WriteComponentResourceFile( instance:TObject;FileName:string;StoreFormAsVisible:Boolean):Boolean';
18492: Function ReadComponentsResourceFile( Components : array of TComponent; FileName : string; NamingMethod :
  TResourceNaming; bLoadTotalForm : Boolean ) : Boolean';
18493: Function WriteComponentsResourceFile( Components : array of TComponent; FileName : string; NamingMethod :
  TResourceNaming ) : Boolean';
18494: Function ReadComponentResourceHeader( sHeaderInfo : TStrings; pSize : Integer; FileName : string;
  NamingMethod : TResourceNaming ) : Boolean';
18495: Function ConvertComponentResourceToTextFile(SourceFileName:string; TargetFileName : string;
  bStoreResNames : Boolean ) : Boolean';
18496: Function
  CheckComponentInResourceFile(Component:TComponent;FileName:string;NamingMethod:TResourceNaming):Boolean;
18497: Function DeleteComponentFromResourceFile( ResourceName : string; FileName : string ) : Boolean';
18498: Function DeleteComponentFromResourceFile8( Component : TComponent; FileName : string; NamingMethod :
  TResourceNaming ) : Boolean';
18499: //CL.AddDelphiFunction('Function CreateFormFromResFile( AOwner : TComponent; NewClassType : TFormClass;
  FileName : string ) : Pointer');
18500: //CL.AddDelphiFunction('Function CreateComponentFromResFile( AOwner : TComponent; NewClassType :
  TComponentClass; FileName : string ) : Pointer');
18501: end;
18502:
18503: procedure SIRegister_SMScript(CL: TPSPascalCompiler);
18504: begin
18505:   CL.FindClass('TOBJECT','TSMSScriptExecutor');
18506:   SIRegister_TSMSError(CL);
18507:   CL.AddClassN(CL.FindClass('TOBJECT','TSMSEModule'));
18508:   CL.AddTypeS('TSMSEProcedureType', '( ptProcedure, ptFunction )');
18509:   SIRegister_TSMSEProcedure(CL);
18510:   SIRegister_TSMSEProcedures(CL);
18511:   SIRegister_TSMSEModule(CL);
18512:   SIRegister_TSMSEModules(CL);
18513:   CL.AddTypeS('TSMSScriptLanguage', '( slCustom, slVBScript, slJavaScript )');
18514:   SIRegister_TSMSScriptExecutor(CL);
18515: //CL.AddDelphiFunction('Procedure Register');
18516: end;
18517:
18518: procedure SIRegister_geometry2(CL: TPSPascalCompiler);
18519: begin
18520:   TPointF2', 'record X : Double; Y : Double; end');
18521:   TPoint3', 'record X : integer; Y : integer; Z : Integer; end');
18522: //CL.AddTypeS('PPointF3', '^TPointF3 // will not work');
18523:   TPointF3', 'record X : double; Y : double; Z : Double; end');
18524: Function AreLinesParallel( p1, p2, p3, p4 : TPoint ) : Boolean';
18525: Function AreLinesParallel1( p1, p2, p3, p4 : TPointF2; e : Double ) : Boolean';
18526: Function AreLinesParallel2( p1, p2, p3, p4, p5, p6 : TPoint3 ) : Boolean';
18527: Function AreLinesParallel3( p1, p2, p3, p4, p5, p6 : TPointF3; e : Double ) : Boolean';
18528: Function IntersectLines( p1, p2, p3, p4 : TPoint ) : TPoint';
18529: Function IntersectLines1( p1, p2, p3, p4 : TPointF2; e : Double ) : TPointF2';
18530: Function AngleDifference( alpha, beta : Double ) : Double';
18531: end;
18532:
18533:
18534: function TRestRequest_createStringStreamFromStringList(strings: TStringList): TStringStream;
18535:
18536: {A simple Oscilloscope using TWaveIn class.
18537: More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
18538: http://www.retroarchive.org/garbo/pc/turbopas/index.html
18539: uses
18540:   Forms,
18541:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
18542:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
18543:   uColorFunctions in 'uColorFunctions.pas',
18544:   AMixer in 'AMixer.pas',
18545:   uSettings in 'uSettings.pas',
18546:   UWavein4 in 'UWavein4.pas',
18547:   USpectrum4 in 'USpectrum4.pas' {Form2},
18548:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
18549:
18550: Functions_max hex in the box maxbox
18551: functionslist.txt
18552: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98/100/101/110/120/160/180/190/192
18553:
18554: ****
18555: Procedure
18556: PROCEDURE SIZE 8875 8396 8289 8242 7507 7401 6792 6310 5971 4438 3797 3600
18557: Procedure *****Now the Procedure list*****
18558: Procedure ( ACol, ARow : Integer; Items : TStrings )
18559: Procedure ( Agg : TAggregate )
18560: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus )
18561: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage )
18562: Procedure ( ASender : TComponent; var AMsg : TIdMessage )

```

```

18563: Procedure ( ASender : TObject; const ABytes : Integer)
18564: Procedure ( ASender : TObject; VStream : TStream)
18565: Procedure ( AThread : TIdThread)
18566: Procedure ( AWebModule : TComponent)
18567: Procedure ( Column : TColumn)
18568: Procedure ( const Username : String; const Password : String; AAuthenticationResult : Boolean)
18569: Procedure ( const iStart : integer; const sText : string)
18570: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
18571: Procedure ( Database : TDatabase; LoginParams : TStrings)
18572: Procedure ( DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:TReconcileAction)
18573: Procedure ( DATASET : TDATASET)
18574: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
18575: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
18576: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
18577: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
18578: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
18579: Procedure ( Done : Integer)
18580: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
18581: Procedure ( HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
18582: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
18583: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
18584: Procedure ( HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
18585: Procedure ( HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
18586: Procedure ( Sender : TCustomListView;const ARect : TRect;Stage:TCustomDrawStage;var DefaultDraw : Boolean)
18587: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
18588: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TownerDrawState)
18589: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
18590: Procedure ( SENDER : TFIELD; const TEXT : String)
18591: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
18592: Procedure ( Sender : TIdTelnet; const Buffer : String)
18593: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
18594: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
18595: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18596: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
18597: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
18598: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
18599: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
18600: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
18601: Procedure ( Sender : TObject; Button : TMPBnType)
18602: Procedure ( Sender : TObject; Button : TMPBnType; var DoDefault : Boolean)
18603: Procedure ( Sender : TObject; Button : TUDBnType)
18604: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
18605: Procedure ( Sender : TObject; ClientSocket : TServerClientWinSocket; var SocketThread : TServerClientThread)
18606: Procedure ( Sender : TObject; Column : TListColumn)
18607: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
18608: Procedure ( Sender : TObject; Connecting : Boolean)
18609: Procedure ( Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var DoneDraw:Bool)
18610: Procedure ( Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
18611: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
18612: Procedure ( Sender : TObject; const UserString:string;var DateAndTime:TDateAndTime;var AllowChange:Boolean)
18613: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
18614: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
18615: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
18616: Procedure ( Sender : TObject; Index : LongInt)
18617: Procedure ( Sender : TObject; Item : TListItem)
18618: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
18619: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
18620: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
18621: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
18622: Procedure ( Sender : TObject; Item : TListItem; var S : string)
18623: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
18624: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
18625: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
18626: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
18627: Procedure ( Sender : TObject; Node : TTTreeNode)
18628: Procedure ( Sender : TObject; Node : TTTreeNode; var AllowChange : Boolean)
18629: Procedure ( Sender : TObject; Node : TTTreeNode; var AllowCollapse : Boolean)
18630: Procedure ( Sender : TObject; Node : TTTreeNode; var AllowEdit : Boolean)
18631: Procedure ( Sender : TObject; Node : TTTreeNode; var AllowExpansion : Boolean)
18632: Procedure ( Sender : TObject; Node : TTTreeNode; var S : string)
18633: Procedure ( Sender : TObject; Node1, Node2 : TTTreeNode; Data : Integer; var Compare : Integer)
18634: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
18635: Procedure ( Sender : TObject; Rect : TRect)
18636: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
18637: Procedure ( Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int)
18638: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
18639: Procedure ( Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
18640: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
18641: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
18642: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
18643: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
18644: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
18645: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
18646: Procedure ( Sender : TObject; Thread : TServerClientThread)
18647: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
18648: Procedure ( Sender : TObject; Username, Password : string)
18649: Procedure ( Sender : TObject; var AllowChange : Boolean)

```

```

18650: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
18651: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
18652: Procedure ( Sender : TObject; var Continue : Boolean)
18653: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMETHOD:TIdHTTPMethod)
18654: Procedure ( Sender : TObject; var Username : string)
18655: Procedure ( Sender : TObject; Wnd : HWND)
18656: Procedure ( Sender : TToolbar; Button : TToolButton)
18657: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
18658: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
18659: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
18660: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
18661: Procedure ( StatusBar : TCustomStatusbar; Panel : TStatusPanel; const Rect : TRect)
18662: Procedure ( StatusBar : TStatusbar; Panel : TStatusPanel; const Rect : TRect)
18663: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
18664: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
18665: procedure (Sender: TObject)
18666: procedure (Sender: TObject; var Done: Boolean)
18667: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
18668: procedure _T(Name: tbtString; v: Variant);
18669: Procedure AbandonSignalHandler( RtlSigNum : Integer)
18670: Procedure Abort
18671: Procedure About1Click( Sender : TObject)
18672: Procedure Accept( Socket : TSocket)
18673: Procedure AESEncryptExecute(const plaintext, ciphertext, password: string)
18674: Procedure AESDecryptFile(const plaintext, ciphertext, password: string)
18675: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
18676: Procedure AESDecryptString(const plaintext: string; var ciphertext: string; password: string)
18677: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
18678: Procedure Add( Addend1, Addend2 : TMyBigInt)
18679: Procedure ADD( const AKEY, AVALUE : VARIANT)
18680: Procedure Add( const Key : string; Value : Integer)
18681: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
18682: Procedure ADD( FIELD : TFIELD)
18683: Procedure ADD( ITEM : TMENUITEM)
18684: Procedure ADD( POPUP : TPOPUPMENU)
18685: Procedure AddCharacters( xCharacters : TCharSet)
18686: Procedure AddDriver( const Name : string; List : TStrings)
18687: Procedure AddImages( Value : TCustomImageList)
18688: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
18689: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
18690: Procedure AddLoader( Loader : TBitmapLoader)
18691: Procedure ADDPARAM( VALUE : TPARAM)
18692: Procedure AddPassword( const Password : string)
18693: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
18694: Procedure AddState( oState : TniRegularExpressionState)
18695: Procedure AddStrings( Strings : TStrings);
18696: procedure AddStrings(Strings: TStrings);
18697: Procedure AddStrings1( Strings : TWideStrings);
18698: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
18699: Procedure AddToRecentDocs( const Filename : string)
18700: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
18701: Procedure AllFunctionsList1Click( Sender : TObject)
18702: procedure AllObjectsList1Click(Sender: TObject);
18703: Procedure Allocate( AAllocateBytes : Integer)
18704: procedure AllResourceList1Click(Sender: TObject);
18705: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
18706: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
18707: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
18708: Procedure AnsiFree( var s : AnsiString)
18709: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
18710: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
18711: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
18712: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
18713: Procedure AntiFreeze;
18714: Procedure APPEND
18715: Procedure Append( const S : WideString)
18716: procedure Append(S: string);
18717: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
18718: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
18719: Procedure AppendChunk( Val : OleVariant)
18720: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
18721: Procedure AppendStr( var Dest : string; S : string)
18722: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
18723: Procedure ApplyRange
18724: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18725: Procedure Arrange( Code : TListArrangement)
18726: procedure Assert(expr : Boolean; const msg: string);
18727: procedure Assert2(expr : Boolean; const msg: string);
18728: Procedure Assign( AList : TCustomBucketList)
18729: Procedure Assign( Other : TObject)
18730: Procedure Assign( Source : TDragObject)
18731: Procedure Assign( Source : TPersistent)
18732: Procedure Assign(Source: TPersistent)
18733: procedure Assign2(mystring, mypath: string);
18734: Procedure AssignCurValues( Source : TDataSet);
18735: Procedure AssignCurValues1( const CurValues : Variant);
18736: Procedure ASSIGNFIELD( FIELD : TFIELD)
18737: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
18738: Procedure AssignFile(var F: Text; FileName: string)

```

```

18739: procedure AssignFile(var F: TextFile; FileName: string)
18740: procedure AssignFileRead(var mystring, myfilename: string);
18741: procedure AssignFileWrite(mystring, myfilename: string);
18742: Procedure AssignTo( Other : TObject)
18743: Procedure AssignValues( Value : TParameters)
18744: Procedure ASSIGNVALUES( VALUE : TPARAMS)
18745: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
18746: Procedure Base64_to_stream( const Base64 : ansiString; Destin : TStream)
18747: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
18748: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
18749: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18750: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
18751: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
18752: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
18753: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
18754: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
18755: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
18756: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
18757: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
18758: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18759: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
18760: procedure Beep
18761: Procedure Beepok
18762: Procedure BeepQuestion
18763: Procedure BeepHand
18764: Procedure BeepExclamation
18765: Procedure BeepAsterisk
18766: Procedure BeepInformation
18767: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
18768: Procedure BeginLayout
18769: Procedure BeginTimer( const Delay, Resolution : Cardinal)
18770: Procedure BeginUpdate
18771: procedure BeginUpdate;
18772: procedure BigScreen1Click(Sender: TObject);
18773: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
18774: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
18775: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
18776: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
18777: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
18778: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
18779: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
18780: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
18781: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
18782: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
18783: Procedure BreakPointMenuClick( Sender : TObject)
18784: procedure BRINGTOFRONT
18785: procedure BringToFront;
18786: Procedure btnBackClick( Sender : TObject)
18787: Procedure btnBrowseClick( Sender : TObject)
18788: Procedure BtnClick( Index : TNavigateBtn)
18789: Procedure btnLargeIconsClick( Sender : TObject)
18790: Procedure BuildAndSendRequest( AURI : TIdURI)
18791: Procedure BuildCache
18792: Procedure BurnMemory( var Buff, BuffLen : integer)
18793: Procedure BurnMemoryStream( Destructo : TMemoryStream)
18794: Procedure CalculateFirstSet
18795: Procedure Cancel
18796: procedure CancelDrag;
18797: Procedure CancelEdit
18798: procedure CANCELHINT
18799: Procedure CancelRange
18800: Procedure CancelUpdates
18801: Procedure CancelWriteBuffer
18802: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIIsRFCMessage:Bool;
18803: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIIsRFCMessage : Boolean);
18804: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIIsRFCMessage:Bool
18805: procedure CaptureScreenFormat(vname: string; vextension: string);
18806: procedure CaptureScreenPNG(vname: string);
18807: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
18808: procedure CASCADE
18809: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
18810: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
18811: Procedure cbPathClick( Sender : TObject)
18812: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18813: Procedure cedbugAfterExecute( Sender : TPSScript)
18814: Procedure cedbugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18815: Procedure cedbugCompile( Sender : TPSScript)
18816: Procedure cedbugExecute( Sender : TPSScript)
18817: Procedure cedbugIdle( Sender : TObject)
18818: Procedure cedbugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18819: Procedure CenterHeight( const pc, pcParent : TControl)
18820: Procedure CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
18821: Procedure CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
18822: Procedure Change
18823: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
18824: Procedure Changed
18825: Procedure ChangeDir( const ADirName : string)
18826: Procedure ChangeDirUp
18827: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)

```

```

18828: Procedure ChangeLevelBy( Value : TChangeRange)
18829: Procedure ChDir(const s: string)
18830: Procedure Check(Status: Integer)
18831: Procedure CheckCommonControl( CC : Integer)
18832: Procedure CHECKFIELDNAME( const FIELDNAME : String)
18833: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
18834: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
18835: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
18836: Procedure CheckToken( T : Char)
18837: procedure CheckToken(t:char)
18838: Procedure CheckTokenSymbol( const S : string)
18839: procedure CheckTokenSymbol(s:string)
18840: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
18841: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18842: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
18843: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
18844: procedure CipherFile1Click(Sender: TObject);
18845: Procedure Clear;
18846: Procedure Clear1Click( Sender : TObject)
18847: Procedure ClearColor( Color : TColor)
18848: Procedure CLEARITEM( AITEM : TMENUITEM)
18849: Procedure ClearMapping
18850: Procedure ClearSelection( KeepPrimary : Boolean)
18851: Procedure ClearWriteBuffer
18852: Procedure Click
18853: Procedure Close
18854: Procedure Close1Click( Sender : TObject)
18855: Procedure CloseDatabase( Database : TDatabase)
18856: Procedure CloseDataSets
18857: Procedure CloseDialog
18858: Procedure CloseFile(var F: Text);
18859: Procedure Closure
18860: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18861: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18862: Procedure CodeCompletionList1Click( Sender : TObject)
18863: Procedure ColEnter
18864: Procedure Collapse
18865: Procedure Collapse( Recurse : Boolean)
18866: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
18867: Procedure CommaSeparatedToStringList( Alist: TStringList; const Value : string)
18868: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
18869: Procedure Compile1Click( Sender : TObject)
18870: procedure ComponentCount1Click(Sender: TObject);
18871: Procedure Compress(azipfolder, azipfile: string)
18872: Procedure DeCompress(azipfolder, azipfile: string)
18873: Procedure XZip(azipfolder, azipfile: string)
18874: Procedure XUnZip(azipfolder, azipfile: string)
18875: Procedure Connect( const ATimeout : Integer)
18876: Procedure Connect( Socket : TSocket)
18877: procedure Console1Click(Sender: TObject);
18878: Procedure Continue
18879: Procedure ContinueCount( var Counter : TJclCounter)
18880: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
18881: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
18882: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
18883: Procedure ConvertImage(vsource, vdestination: string);
18884: // Ex. ConvertImage(Exepath+'my233.bmp.bmp',Exepath+'mypng111.png')
18885: Procedure ConvertBitmap(vsource, vdestination: string);
18886: Procedure ConvertToGray(Cnv: TCanvas);
18887: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
18888: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
18889: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
18890: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
18891: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
18892: Procedure CopyFrom( mbCopy : TMyBigInt)
18893: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
18894: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
18895: Procedure CopyTidByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array of Byte; const ADestIndex : Integer; const ALength : Integer)
18896: Procedure CopyTidBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const ALen:Int )
18897: Procedure CopyTidCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
18898: Procedure CopyTidInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
18899: Procedure CopyTidIPv6Address(const ASource:TidIPv6Address; var VDest : TIdBytes; const ADestIndex : Integer)
18900: Procedure CopyTidLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
18901: Procedure CopyTidNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
18902: Procedure CopyTidNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18903: Procedure CopyTidString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
18904: Procedure CopyTidWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18905: Procedure CopyToClipboard
18906: Procedure CountParts
18907: Procedure CreateDataSet
18908: Procedure CreateEmptyFile( const FileName : string)
18909: Procedure CreateFileFromString( const FileName, Data : string)
18910: Procedure CreateFromDelta( Source : TPacketDataSet)
18911: procedure CREATEHANDLE
18912: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
18913: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)

```

```

18914: Procedure CreateProcAsUserEx( const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar )
18915: Procedure CreateTable
18916: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString )
18917: procedure CSyntax1Click(Sender: TObject);
18918: Procedure CurrencyToComp( Value : Currency; var Result : Comp )
18919: Procedure CURSORPOSCHANGED
18920: procedure CutFirstDirectory(var S: String)
18921: Procedure DataBaseError(const Message: string)
18922: Procedure DateTimeToString( var Result: string; Format : string; DateTime : TDateTime );
18923: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
18924: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime )
18925: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
18926: Procedure DBIError(errorCode: Integer)
18927: Procedure DebugOutput( const AText : string )
18928: procedure DebugIn(DebugLOGFILE: string; Event_message: string );
18929: Procedure DebugRun1Click( Sender : TObject )
18930: procedure Dec;
18931: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word )
18932: procedure DecodeDate( const DateTime: TDateTime; var Year, Month, Day: Word );
18933: Procedure DecodeDateDay( const AValue: TDateTime; out AYear, ADayOfYear : Word )
18934: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekofMonth,ADayOfWeek : Word)
18935: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AMin,ASec,AMillSec:Word)
18936: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word )
18937: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
18938: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word )
18939: procedure DecodeTime( const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word );
18940: Procedure Decompile1Click( Sender : TObject )
18941: Procedure DefaultDrawColumnCell( const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState )
18942: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState )
18943: Procedure DeferLayout
18944: Procedure deffileread
18945: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
18946: Procedure DelayMicroseconds( const MicroSeconds : Integer )
18947: Procedure Delete
18948: Procedure Delete( const Afilename : string )
18949: Procedure Delete( const Index : Integer )
18950: Procedure DELETE( INDEX : INTEGER )
18951: Procedure Delete( Index : LongInt )
18952: Procedure Delete( Node : TTreenode )
18953: procedure Delete(var s: AnyString; ifrom, icount: Longint );
18954: Procedure DeleteAlias( const Name : string )
18955: Procedure DeleteDriver( const Name : string )
18956: Procedure DeleteIndex( const Name : string )
18957: Procedure DeleteKey( const Section, Ident : String )
18958: Procedure DeleteRecords
18959: Procedure DeleteRecords( AffectRecords : TAffectRecords )
18960: Procedure DeleteString( var pStr : String; const pDelStr : string )
18961: Procedure DeleteTable
18962: procedure DelphiSite1Click(Sender: TObject);
18963: Procedure Deselect
18964: Procedure Deselect( Node : TTreenode )
18965: procedure DestroyComponents
18966: Procedure DestroyHandle
18967: Procedure Diff( var X : array of Double )
18968: procedure Diff(var X: array of Double);
18969: Procedure DirCreate( const DirectoryName : String )';
18970: procedure DISABLEALIGN
18971: Procedure DisableConstraints
18972: Procedure Disconnect
18973: Procedure Disconnect( Socket : TSocket )
18974: Procedure Dispose
18975: procedure Dispose(P: PChar)
18976: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word )
18977: Procedure DoKey( Key : TDBCtrlGridKey )
18978: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreenode; aTreeView: TTreerView);
18979: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreenode; aTreeView: TTreerView);
18980: Procedure Dormant
18981: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd );
18982: Procedure DoubleToBytes( Value : Double; Bytes : array of byte )
18983: Procedure DoubleToComp( Value : Double; var Result : Comp )
18984: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
18985: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean );
18986: procedure Draw(X, Y: Integer; Graphic: TGraphic);
18987: Procedure Draw(Canvas:TCanvas; X,Y,
Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
18988: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer )
18989: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean )
18990: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer )
18991: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState )
18992: procedure DrawFocusRect(const Rect: TRect);
18993: Procedure DrawHBITToTBitmap( HDIB : THandle; Bitmap : TBitmap )
18994: Procedure DRAWMENUIITEM( MENUIITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE )
18995: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Int;ImageIndex:Int;Overlay:TOverlay;Enabled:Boolean);
18996: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle : TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
18997: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
18998: Procedure DrawPolyline( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect )
18999: Procedure DropConnections
19000: Procedure DropDown

```

```

19001: Procedure DumpDescription( oStrings : TStrings )
19002: Procedure DumpStateTable( oStrings : TStrings )
19003: Procedure EDIT
19004: Procedure EditButtonClick
19005: Procedure EditFont1Click( Sender : TObject )
19006: procedure Ellipse(X1, Y1, X2, Y2: Integer);
19007: Procedure Ellipse( const Rect : TRect );
19008: Procedure EMMS
19009: Procedure Encode( ADest : TStream )
19010: procedure ENDDRAG(DROP:BOOLEAN)
19011: Procedure EndEdit( Cancel : Boolean )
19012: Procedure EndTimer
19013: Procedure EndUpdate
19014: Procedure EraseSection( const Section : string )
19015: Procedure Error( const Ident : string )
19016: procedure Error(Ident:Integer)
19017: Procedure ErrorFmt( const Ident : string; const Args : array of const )
19018: Procedure ErrorStr( const Message : string )
19019: procedure ErrorStr(Message:String)
19020: Procedure Exchange( Index1, Index2 : Integer )
19021: procedure Exchange(Index1, Index2: Integer);
19022: Procedure Exec( FileName, Parameters, Directory : string )
19023: Procedure ExecProc
19024: Procedure ExecSQL( UpdateKind : TUpdateKind )
19025: Procedure Execute
19026: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant )
19027: Procedure ExecuteAndWait( FileName : string; Visibility : Integer )
19028: Procedure ExecuteCommand(executeFile, paramstring: string)
19029: Procedure ExecuteShell(executeFile, paramstring: string)
19030: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
19031: Procedure ExitThread(ExitCode: Integer); stdcall;
19032: Procedure ExitProcess(ExitCode: Integer); stdcall;
19033: Procedure Expand( AUserName : String; AResults : TStrings )
19034: Procedure Expand( Recurse : Boolean )
19035: Procedure ExportClipboardClick( Sender : TObject )
19036: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress )
19037: Procedure ExtractContentFields( Strings : TStrings )
19038: Procedure ExtractCookieFields( Strings : TStrings )
19039: Procedure ExtractFields( Separators,WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings )
19040: Procedure ExtractHeaderFields(Separ,
 WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
19041: Procedure ExtractHTTPFields(Separators,WhiteSpace:
  TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
19042: Procedure ExtractQueryFields( Strings : TStrings )
19043: Procedure FastDegToGrad
19044: Procedure FastDegToRad
19045: Procedure FastGradToDeg
19046: Procedure FastGradToRad
19047: Procedure FastRadToDeg
19048: Procedure FastRadToGrad
19049: Procedure FileClose( Handle : Integer )
19050: Procedure FileClose(handle: integer)
19051: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs>ShowDirs,Multitasking:Bool)
19052: Procedure Filestructure( AStructure : TidFTPDataStructure )
19053: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
19054: Procedure FillBytes( var VBytes : TidBytes; const ACount : Integer; const AValue : Byte )
19055: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte )
19056: Procedure FillChar2(var X: PChar ; count: integer; value: char)
19057: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
19058: Procedure FillIPList
19059: procedure FillRect(const Rect: TRect );
19060: Procedure FillTStrings( AStrings : TStrings )
19061: Procedure FilterOnBookmarks( Bookmarks : array of const )
19062: procedure FinalizePackage(Module: HMODULE)
19063: procedure FindClose;
19064: procedure FindClose2(var F: TSearchRec)
19065: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent );
19066: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
19067: Procedure FindNearest( const KeyValues : array of const )
19068: Procedure FinishContext
19069: Procedure FIRST
19070: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float )
19071: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int );
19072: Procedure FloodPoint( X, Y : Integer; Color : TColor; FillStyle : TFillStyle )
19073: Procedure FlushSchemaCache( const TableName : string )
19074: procedure FmtStr( var Result: string; const Format: string; const Args: array of const )
19075: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
19076: Procedure Form1Close( Sender : TObject; var Action : TCloseAction )
19077: Procedure FormActivate( Sender : TObject )
19078: procedure FormatLn(const format: String; const args: array of const); //alias
19079: Procedure FormClose( Sender : TObject; var Action : TCloseAction )
19080: Procedure FormCreate( Sender : TObject )
19081: Procedure FormDestroy( Sender : TObject )
19082: Procedure FormKeyPress( Sender : TObject; var Key : Char )
19083: procedure FormOutput1Click(Sender: TObject);
19084: Procedure FormToHtml( Form : TForm; Path : string )
19085: procedure FrameRect(const Rect: TRect );
19086: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer )
19087: Procedure NotebookHandlesNeeded( Notebook : TNotebook )

```

```

19088: Procedure Free( Buffer : TRecordBuffer)
19089: Procedure Free( Buffer : TValueBuffer)
19090: Procedure Free;
19091: Procedure FreeAndNil(var Obj:TObject)
19092: Procedure FreeImage
19093: procedure FreeMem(P: PChar; Size: Integer)
19094: Procedure FreeTreeData( Tree : TUpdateTree)
19095: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
19096: Procedure FullCollapse
19097: Procedure FullExpand
19098: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short); //InterBase
19099: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
19100: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
19101: Procedure Get1( AURL: string; const AResponseContent : TStream);
19102: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
19103: Procedure Get2(const ASourcefile,ADestfile: string;const ACanOverwrite: boolean; AResume: Boolean);
19104: Procedure GetAliasNames( List : TStrings)
19105: Procedure GetAliasParams( const AliasName : string; List : TStrings)
19106: Procedure GetApplicationsRunning( Strings : TStrings)
19107: Procedure getBox(aURL, extension: string);
19108: Procedure GetCommandTypes( List : TWideStrings)
19109: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
19110: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
19111: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
19112: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
19113: Procedure GetDatabaseNames( List : TStrings)
19114: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
19115: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
19116: Procedure GetDir(d: byte; var s: string)
19117: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
19118: Procedure GetDriverNames( List : TStrings)
19119: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
19120: Procedure GetDriverParams( const DriverName : string; List : TStrings)
19121: Procedure GetEmails1Click( Sender : TObject)
19122: Procedure getEnvironmentInfo;
19123: Function getEnvironmentString: string;
19124: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
19125: Procedure GetFieldNames( const TableName : string; List : TStrings)
19126: Procedure GetFieldNames( const TableName : string; List : TStrings);
19127: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
19128: Procedure GETFIELDNAMES( LIST : TSTRINGS)
19129: Procedure GetFieldNames1( const TableName : string; List : TStrings);
19130: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
19131: Procedure GetFieldNames2( const TableName : WideString; SchemaName : WideString; List : TWideStrings);
19132: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
19133: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
19134: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
19135: Procedure GetFMTBCD( Buffer : TRecordBuffer; var value : TBCD)
19136: Procedure GetFormatSettings
19137: Procedure GetFromDIB( var DIB : TBitmapInfo)
19138: Procedure GetFromHDI( HDI : HBitmap)
19139: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
19140: Procedure GetGEOMap(C_form,apath: string; const Data: string); //C_form: [html/json/xml]
19141: Procedure GetIcon( Index : Integer; Image : TIcon);
19142: Procedure GetIcon1(Index : Integer; Image: TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
19143: Procedure GetIndexInfo( IndexName : string)
19144: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
19145: Procedure GetIndexNames( List : TStrings)
19146: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
19147: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
19148: Procedure GetIndexNames4( const TableName : string; List : TStrings);
19149: Procedure GetInternalResponse
19150: Procedure GETITEMNAMES( LIST : TSTRINGS)
19151: procedure GetMem(P: PChar; Size: Integer)
19152: Procedure GETOLE2ACCELERORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
19153: procedure GetPackageDescription(ModuleName: PChar): string)
19154: Procedure GetPackageName( List : TStrings);
19155: Procedure GetPackageNames1( List : TWideStrings);
19156: Procedure GetParamList( List : TList; const ParamNames : WideString)
19157: Procedure GetProcedureNames( List : TStrings);
19158: Procedure GetProcedureNames( List : TWideStrings);
19159: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
19160: Procedure GetProcedureNames1( List : TStrings);
19161: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
19162: Procedure GetProcedureNames3( List : TWideStrings);
19163: Procedure GetProcedureNames4( const PackageName : WideString; List : TWideStrings);
19164: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
19165: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
19166: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
19167: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : WideString; List : TList);
19168: Procedure GetProviderNames( Names : TWideStrings);
19169: Procedure GetProviderNames( Proc : TGetStrProc)
19170: Procedure GetProviderNames1( Names : TStrings);
19171: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
19172: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string); //no open image
19173: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const Data:string;apath:string);
19174: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
19175: Procedure GetSchemaNames( List : TStrings);

```

```

19176: Procedure GetSchemaNames1( List : TWideStrings );
19177: Procedure getScriptandRunAsk;
19178: Procedure getScriptandRun(ascript: string);
19179: Procedure getScript(ascript: string); //alias
19180: Procedure getWebScript(ascript: string); //alias
19181: Procedure GetSessionNames( List : TStrings )
19182: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings )
19183: Procedure GetStrings( List : TStrings )
19184: Procedure GetSystemTime; stdcall;
19185: Procedure GetTableNames(const DatabaseName, Pattern:string;Extensions, SystemTables:Boolean;List:TStrings)
19186: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
19187: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
19188: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
19189: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
19190: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
19191: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
19192: Procedure GetTransitionsOn( cChar : char; oStateList : TList )
19193: Procedure GetVisibleWindows( List : TStrings )
19194: Procedure GoBegin
19195: Procedure GotoCurrent( DataSet : TCustomClientDataSet )
19196: Procedure GotoCurrent( Table : TTable )
19197: procedure GotoEnd1Click(Sender: TObject);
19198: Procedure GotoNearest
19199: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
Direction: TGradientDirection)
19200: Procedure HandleException( E : Exception; var Handled : Boolean )
19201: procedure HANDLEMESSAGE
19202: procedure HandleNeeded;
19203: Procedure Head( AURL : string )
19204: Procedure Help( var AHelpContents : TStringList; ACommand : String )
19205: Procedure HexToBinary( Stream : TStream )
19206: procedure HexToBinary(Stream:TStream)
19207: Procedure HideDragImage
19208: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean )
19209: Procedure HideTraybar
19210: Procedure HideWindowForSeconds(secs: integer); //3 seconds
19211: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); //3 seconds
19212: Procedure HookOSExceptions
19213: Procedure HookSignal( RtlSigNum : Integer )
19214: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single );
19215: Procedure HTMLOnSyntax1Click( Sender : TObject )
19216: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPPSPascalCompiler )
19217: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSEexec; x : TPSRuntimeClassImporter )
19218: Procedure ImportFromClipboard1Click( Sender : TObject )
19219: Procedure ImportFromClipboard2Click( Sender : TObject )
19220: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer )
19221: procedure Incb(var x: byte);
19222: Procedure IncludelClick( Sender : TObject )
19223: Procedure IncludeOFF; //preprocessing
19224: Procedure IncludeON;
19225: procedure Info1Click(Sender: TObject);
19226: Procedure InitAltRecBuffers( CheckModified : Boolean )
19227: Procedure InitContext( Request : TWebRequest; Response : TWebResponse )
19228: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse )
19229: Procedure InitData( ASource : TDataSet )
19230: Procedure InitDelta( ADelta : TPacketDataSet );
19231: Procedure InitDelta1( const ADelta : OleVariant );
19232: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse )
19233: Procedure Initialize
19234: procedure InitializePackage(Module: HMODULE)
19235: Procedure INITIACTION
19236: Procedure initHexArray(var hexn: THexArray); //THexArray', 'array[0..15] of char,'
19237: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet )
19238: Procedure InitModule( AModule : TComponent )
19239: Procedure InitStdConvs
19240: Procedure InitTreeData( Tree : TUpdateTree )
19241: Procedure INSERT
19242: Procedure Insert( Index : Integer; AClass : TClass )
19243: Procedure Insert( Index : Integer; AComponent : TComponent )
19244: Procedure Insert( Index : Integer; AObject : TObject )
19245: Procedure Insert( Index : Integer; const S : WideString )
19246: Procedure Insert( Index : Integer; Image, Mask : TBitmap )
19247: Procedure Insert(Index: Integer; const S: string);
19248: procedure Insert(Index: Integer; S: string);
19249: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
19250: procedure InsertComponent(AComponent:TComponent)
19251: procedure InsertControl(AControl: TControl);
19252: Procedure InsertIcon( Index : Integer; Image : TIcon )
19253: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor )
19254: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject )
19255: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
19256: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte )
19257: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte )
19258: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte )
19259: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
19260: Procedure InternalBeforeResolve( Tree : TUpdateTree )
19261: Procedure InvalidateModuleCache
19262: Procedure InvalidateTitles
19263: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word )

```

```

19264: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
19265: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
  ABaseDate:TDateTime)
19266: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
19267: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
19268: procedure JavaSyntax1Click(Sender: TObject);
19269: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
19270: Procedure KillDataChannel
19271: Procedure Largefont1Click( Sender : TObject)
19272: Procedure LAST
19273: Procedure LaunchCpl( FileName : string)
19274: Procedure Launch( const AFile : string)
19275: Procedure LaunchFile( const AFile : string)
19276: Procedure LetfileList(FileList: TStringlist; apath: string);
19277: Procedure lineToNumber( xmemo : String; met : boolean)
19278: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
  DefaultDraw:Bool)
19279: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
  : TCustDrawState; var DefaultDraw : Boolean)
19280: Procedure ListViewData( Sender : TObject; Item : TListItem)
19281: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
  : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
19282: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
19283: Procedure ListViewDblClick( Sender : TObject)
19284: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
19285: Procedure ListDLEExports(const FileName: string; List: TStrings);
19286: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
19287: procedure LoadBytecode1Click(Sender: TObject);
19288: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
19289: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
19290: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
19291: Procedure LoadFromFile( AFileName : string)
19292: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
19293: Procedure LoadFromFile( const FileName : string)
19294: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
19295: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
19296: Procedure LoadFromFile( const FileName : WideString)
19297: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
19298: Procedure LoadFromFile(const AFileName: string)
19299: procedure LoadFromFile(FileName: string);
19300: procedure LoadFromFile(FileName:String)
19301: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
19302: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
19303: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
19304: Procedure LoadFromStream( const Stream : TStream)
19305: Procedure LoadFromStream( S : TStream)
19306: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19307: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19308: Procedure LoadFromStream( Stream : TStream)
19309: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
19310: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
19311: procedure LoadFromStream(Stream: TStream);
19312: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
19313: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
19314: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
19315: Procedure LoadLastfileClick( Sender : TObject)
19316: { LoadIcoToImage loads two icons from resource named NameRes,
  into two image lists ALarge and ASmall}
19317: 
19318: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
19319: Procedure LoadMemo
19320: Procedure LoadParamsFromIniFile( FFileName : WideString)
19321: Procedure Lock
19322: Procedure Login
19323: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
19324: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
19325: Procedure MakeCaseInsensitive
19326: Procedure MakeDeterministic( var bChanged : boolean)
19327: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
19328: // type TVolumeLevel = 0..127; , savaFilePath as C:\MyFile.wav
19329: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
19330: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
19331: Procedure SetComplexSoundElements(freqedt,Phaseadt,AmpEdt,WaveGrp:integer);
19332: Procedure SetRectComplexFormatStr( const S : string)
19333: Procedure SetPolarComplexFormatStr( const S : string)
19334: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
19335: Procedure MakeVisible
19336: Procedure MakeVisible( PartialOK : Boolean)
19337: Procedure Manual1Click( Sender : TObject)
19338: Procedure MarkReachable
19339: Procedure maxBox; //shows the exe version data in a win box
19340: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
19341: Procedure Memo1Change( Sender : TObject)
19342: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
  Action:TSynReplaceAction)
19343: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
19344: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
19345: procedure Memory1Click(Sender : TObject);
19346: Procedure MERGE( MENU : TMAINMENU)
19347: Procedure MergeChangeLog

```

```

19348: procedure MINIMIZE
19349: Procedure MinimizeMaxbox;
19350: procedure MyCopyFile(Name1,Name2:string);
19351: Procedure MkDir(const s: string)
19352: Procedure MakeDir(const s: string)');
19353: Procedure ChangeDir(const s: string)');
19354: Function makeFile(const FileName: string): integer)');
19355: Procedure mnuPrintFont1Click( Sender : TObject)
19356: procedure ModalStarted
19357: Procedure Modified
19358: Procedure ModifyAlias( Name : string; List : TStrings)
19359: Procedure ModifyDriver( Name : string; List : TStrings)
19360: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
19361: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
19362: Procedure Move( CurIndex, NewIndex : Integer)
19363: procedure Move(CurIndex, NewIndex: Integer);
19364: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
19365: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
19366: Procedure moveCube( o : TMyLabel)
19367: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
19368: procedure MoveTo(X, Y: Integer);
19369: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
19370: Procedure MovePoint(var x,y:Extended; const angle:Extended);
19371: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
19372: Procedure Multiplyl( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
19373: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
19374: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
19375: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
19376: procedure New(P: PChar)
19377: procedure New1Click(Sender: TObject);
19378: procedure NewInstance1Click(Sender: TObject);
19379: Procedure NEXT
19380: Procedure NextMonth
19381: Procedure Noop
19382: Procedure NormalizePath( var APath : string)
19383: procedure ObjectBinaryToText(Input, Output: TStream)
19384: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19385: procedure ObjectResourceToText(Input, Output: TStream)
19386: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19387: procedure ObjectTextToBinary(Input, Output: TStream)
19388: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19389: procedure ObjectTextToResource(Input, Output: TStream)
19390: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19391: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
19392: Procedure Open( const UserID : WideString; const Password : WideString);
19393: Procedure Open;
19394: Procedure open1Click( Sender : TObject)
19395: Procedure OpenCdDrive
19396: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
19397: Procedure OpenCurrent
19398: Procedure OpenFile(vfilenamepath: string)
19399: Procedure OpenDirectory1Click( Sender : TObject)
19400: Procedure OpenDir(adir: string);
19401: Procedure OpenIndexFile( const IndexName : string)
19402: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const SchemaID:OleVariant;DataSet:TADODataset)
19403: Procedure OpenWriteBuffer( const AThreshold : Integer)
19404: Procedure OptimizeMem
19405: Procedure Options1( AURL : string);
19406: Procedure OutputDebugString(lpOutputString : PChar)
19407: Procedure PackBuffer
19408: Procedure Paint
19409: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
19410: Procedure PaintToBitmap( Target : TBitmap)
19411: Procedure PaletteChanged
19412: Procedure ParentBidiModeChanged
19413: Procedure PARENTBIDI MODECHANGED( ACONTROL : TOBJECT)
19414: Procedure PasteFromClipboard;
19415: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
19416: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
19417: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
19418: Procedure PError( Text : string)
19419: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
19420: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
19421: Procedure Play(FromFrame, ToFrame : Word; Count : Integer)
19422: procedure playmp3(mpPath: string);
19423: Procedure PlayMP31Click( Sender : TObject)
19424: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
19425: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
19426: procedure PolyBezier(const Points: array of TPoint);
19427: procedure PolyBezierTo(const Points: array of TPoint);
19428: procedure Polygon(const Points: array of TPoint);
19429: procedure Polyline(const Points: array of TPoint);
19430: Procedure Pop
19431: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU; GROUPS: array of INT; var WIDTHS : array of LONGINT)
19432: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
19433: Procedure POPUP( X, Y : INTEGER)
19434: Procedure PopupURL(URL : WideString);
19435: Procedure POST

```

```

19436: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
19437: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
19438: Procedure Post6( AURL : string; const ASource : TIIdMultiPartFormDataStream; AResponseContent : TStream);
19439: Procedure PostUser( const Email, FirstName, LastName : WideString);
19440: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
19441: procedure Pred(X: int64);
19442: Procedure Prepare;
19443: Procedure PrepareStatement;
19444: Procedure PreProcessXML( AList : TStrings);
19445: Procedure PreventDestruction;
19446: Procedure Print( const Caption : string);
19447: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
19448: procedure printf(const format: String; const args: array of const);
19449: Procedure PrintList(Value: TStringList);
19450: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle); //TBitmapStyle=(bsNormal,bsCentered,bsStretched)
19451: Procedure PrintoutClick( Sender : TObject);
19452: Procedure ProcessHeaders;
19453: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR);
19454: Procedure ProcessMessage( AMsg : TIIdMessage; AHeaderOnly : Boolean);
19455: Procedure ProcessMessage1( AMsg : TIIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
19456: Procedure ProcessMessage2( AMsg : TIIdMessage; const AFilename : string; AHeaderOnly : Boolean);
19457: Procedure ProcessMessagesOFF; //application.processmessages
19458: Procedure ProcessMessagesON;
19459: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string);
19460: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
19461: Procedure Prolist Size is: 3797 /1415;
19462: Procedure procMessClick( Sender : TObject);
19463: Procedure PSScriptCompile( Sender : TPSScript);
19464: Procedure PSScriptExecute( Sender : TPSScript);
19465: Procedure PSScriptLine( Sender : TObject);
19466: Procedure Push( ABoundary : string);
19467: procedure PushItem(AItem: Pointer);
19468: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
19469: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
19470: procedure PutLinuxLines(const Value: string);
19471: Procedure Quit;
19472: Procedure RaiseConversionError( const AText : string);
19473: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
19474: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string);
19475: procedure RaiseException(Ex: TIEException; Param: String);
19476: Procedure RaiseExceptionForLastCmdResult;
19477: procedure RaiseLastException;
19478: procedure RaiseException2;
19479: Procedure RaiseException3(const Msg: string);
19480: Procedure RaiseExcept(const Msg: string);
19481: Procedure RaiseLastOSError;
19482: Procedure RaiseLastWin32;
19483: procedure RaiseLastWin32Error;
19484: Procedure RaiseListError( const ATemplate : string; const AData : array of const);
19485: Procedure RandomFillStream( Stream : TMemoryStream);
19486: procedure randomize;
19487: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect);
19488: Procedure RCS;
19489: Procedure Read( Socket : TSocket);
19490: procedure Readln1(var ast: string); //of inputquery;
19491: Procedure ReadblobData;
19492: procedure ReadBuffer(Buffer:String;Count:LongInt);
19493: procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
19494: Procedure ReadSection( const Section : string; Strings : TStrings);
19495: Procedure ReadSections( Strings : TStrings);
19496: Procedure ReadSections( Strings : TStrings);
19497: Procedure ReadSections1( const Section : string; Strings : TStrings);
19498: Procedure ReadSectionValues( const Section : string; Strings : TStrings);
19499: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean);
19500: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer);
19501: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
19502: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
19503: Procedure Realign;
19504: procedure Rectangle(X1, Y1, X2, Y2: Integer);
19505: Procedure Rectangle1( const Rect : TRect);
19506: Procedure RectCopy( var Dest : TRect; const Source : TRect);
19507: Procedure RectFitToScreen( var R : TRect);
19508: Procedure RectGrow( var R : TRect; const Delta : Integer);
19509: Procedure RectGrowX( var R : TRect; const Delta : Integer);
19510: Procedure RectGrowY( var R : TRect; const Delta : Integer);
19511: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer);
19512: Procedure RectMoveTo( var R : TRect; const X, Y : Integer);
19513: Procedure RectNormalize( var R : TRect);
19514: // TFileCallbackProcedure = procedure(filename:string);
19515: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
19516: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
19517: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState);
19518: Procedure Refresh;
19519: Procedure RefreshData( Options : TFetchOptions);
19520: Procedure REFRESHLOOKUPLIST;
19521: Procedure regExPathfinder(Pathin, fileout, firsttp, aregex, ext: string; asort: boolean);
19522: Procedure RegExPathfinder2(Pathin, fileout, firsttp, aregex, ext: string; asort, acopy: boolean);
19523: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIIdAuthenticationClass);
19524: Procedure RegisterChanges( Value : TChangeLink);

```

```

19525: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
19526: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
19527: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
19528: Procedure ReInitialize( ADelay : Cardinal)
19529: procedure RELEASE
19530: Procedure Remove( const AByteCount : integer)
19531: Procedure REMOVE( FIELD : TFIELD)
19532: Procedure REMOVE( ITEM : TMENUITEM)
19533: Procedure REMOVE( POPUP : TPOPUPMENU)
19534: Procedure RemoveAllPasswords
19535: procedure RemoveComponent(AComponent:TComponent)
19536: Procedure RemoveDir( const ADirName : string)
19537: Procedure RemoveLambdaTransitions( var bChanged : boolean)
19538: Procedure REMOVEPARAM( VALUE : TPARAM)
19539: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
19540: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
19541: Procedure Rename( const ASourceFile, ADestFile : string)
19542: Procedure Rename( const FileName : string; Reload : Boolean)
19543: Procedure RenameTable( const NewTableName : string)
19544: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
19545: Procedure ReplaceClick( Sender : TObject)
19546: Procedure ReplaceDate( var Date : TDateTime; NewDate : TDateTime)
19547: procedure ReplaceDate(var Date: TDateTime; const NewDate: TDateTime))
19548: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
19549: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
19550: Procedure ReplaceTime( var Date : TDateTime; NewTime : TDateTime)
19551: procedure ReplaceTime(var Date: TDateTime; const NewTime: TDateTime);
19552: Procedure Requery( Options : TExecuteOptions)
19553: Procedure Reset
19554: Procedure Reset1Click( Sender : TObject)
19555: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
19556: procedure ResourceExplore1Click(Sender: TObject);
19557: Procedure RestoreContents
19558: Procedure RestoreDefaults
19559: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
19560: Procedure RetrieveHeaders
19561: Procedure RevertRecord
19562: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
19563: Procedure RGBAToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19564: Procedure RGBAToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19565: Procedure RGBAToHSL( const R, G, B : Single; out H, S, L : Single);
19566: Procedure RGBAToHSL1( const RGB : TColor32; out H, S, L : Single);
19567: Procedure RGBAToHSV( r, g, b : Integer; var h, s, v : Integer)
19568: Procedure RleCompress2( Stream : TStream)
19569: Procedure RleDecompress2( Stream : TStream)
19570: Procedure RmDir(const S: string)
19571: Procedure Rollback
19572: Procedure Rollback( TransDesc : TTransactionDesc)
19573: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
19574: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
19575: Procedure RollbackTrans
19576: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
19577: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
19578: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
19579: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
19580: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
19581: Procedure S_EBox( const AText : string)
19582: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:int;var AddKey:int)
19583: Procedure S_IBox( const AText : string)
19584: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
19585: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
19586: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
19587: Procedure SampleVarianceAndMean
19588: ( const X : TDynFloatArray; var Variance, Mean : Float)
19589: Procedure Save2Click( Sender : TObject)
19590: Procedure Saveas3Click( Sender : TObject)
19591: Procedure Savebefore1Click( Sender : TObject)
19592: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
19593: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19594: Procedure SaveConfigFile
19595: Procedure SaveOutput1Click( Sender : TObject)
19596: procedure SaveScreenshotClick(Sender: TObject);
19597: Procedure SaveLn(pathname, content: string); //Saveln(exepath+'mysavelntest.txt', memo2.text);
19598: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
19599: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
19600: Procedure SaveToFile( AFileName : string)
19601: Procedure SAVETOFILE( const FILENAME : String)
19602: Procedure SaveToFile( const FileName : WideString)
19603: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
19604: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
19605: procedure SaveToFile(FileName: string);
19606: procedure SaveToFile(FileName:String)
19607: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
19608: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19609: Procedure SaveToStream( S : TStream)
19610: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19611: Procedure SaveToStream( Stream : TStream)
19612: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
19613: procedure SaveToStream(Stream: TStream);

```

```

19614: procedure SaveToStream(Stream:TStream)
19615: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
19616: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
19617: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
19618: procedure Say(const sText: string)
19619: Procedure SBytecode1Click( Sender : TObject)
19620: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
19621: procedure ScriptExplorer1Click(Sender: TObject);
19622: Procedure Scroll( Distance : Integer)
19623: Procedure Scroll( DX, DY : Integer)
19624: procedure ScrollBy(DeltaX, DeltaY: Integer);
19625: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
19626: Procedure ScrollTabs( Delta : Integer)
19627: Procedure Search1Click( Sender : TObject)
19628: procedure SearchAndOpenDoc(vfilenamepath: string)
19629: procedure SearchAndOpenFile(vfilenamepath: string)
19630: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
19631: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19632: Procedure SearchNext1Click( Sender : TObject)
19633: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
19634: Procedure Select1( const Nodes : array of TTreeNode);
19635: Procedure Select2( Nodes : TList);
19636: Procedure SelectNext( Direction : Boolean)
19637: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
19638: Procedure SelfTestPEM //unit uPSI_CPEM
19639: Procedure Send( AMsg : TIdMessage)
19640: //config forst in const MAILINIFILE = 'maildef.ini';
19641: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
19642: Procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
19643: Procedure SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
19644: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
19645: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
19646: Procedure SendResponse
19647: Procedure SendStream( AStream : TStream)
19648: procedure SendMCICmd(Cmd: string); !
19649: Procedure Set8087CW( NewCW : Word)
19650: Procedure SetAll( One, Two, Three, Four : Byte)
19651: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
19652: Procedure SetAppDispatcher( const AD dispatcher : TComponent)
19653: procedure SetArrayLength;
19654: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer); //2 dimension
19655: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19656: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer); //all init
19657: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
19658: procedure SetArrayLength2Char2(var arr: T2CharArray; asize1, asize2: integer);'
19659: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);'
19660: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);'
19661: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
19662: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
19663: Procedure SetAsHandle( Format : Word; Value : THandle)
19664: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
19665: procedure SetCaptureControl(Control: TControl);
19666: Procedure SetColumnAttributes
19667: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
19668: Procedure SetCustomHeader( const Name, Value : string)
19669: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
  FieldName:Widestring)
19670: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
19671: Procedure SetFocus
19672: procedure SetFocus; virtual;
19673: Procedure SetInitialState
19674: Procedure SetKey
19675: procedure SetLastError(ErrorCode: Integer)
19676: procedure SetLength;
19677: procedure SetLength2(var S: string; NewLength: Integer)');
19678: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
19679: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
19680: Procedure SetParams( ADataSet : TDataSet; UpdateKind : TUpdateKind);
19681: procedure SETPARAMS(APosition,AMin,AMax:INTEGER)
19682: Procedure SetParam1( UpdateKind : TUpdateKind);
19683: Procedure SetPassword( const Password : string)
19684: Procedure SetPointer( Ptr : Pointer; Size : Longint)
19685: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
19686: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
19687: Procedure SetProvider( Provider : TComponent)
19688: Procedure SetProxy( const Proxy : string)
19689: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
19690: Procedure SetRange( const StartValues, EndValues : array of const)
19691: Procedure SetRangeEnd
19692: Procedure SetRate( const aPercent, aYear : integer)
19693: procedure SetRate(const aPercent, aYear: integer)
19694: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19695: Procedure SetSafeCallExceptionMsg( Msg : String)
19696: procedure SETSELTEXTBUF(BUFFER:PCHAR)
19697: Procedure SetSize( AWidth, AHeight : Integer)
19698: procedure SetSize(NewSize:LongInt)
19699: procedure SetString(var s: string; buffer: PChar; len: Integer)
19700: Procedure SetStrings( List : TStrings)
19701: Procedure SetText( Text : PwideChar)

```

```

19702: procedure SetText(Text: PChar);
19703: Procedure SetTextBuf( Buffer : PChar)
19704: procedure SETTEXTBUF(BUFFER:PCHAR)
19705: Procedure SetTick( Value : Integer)
19706: Procedure SetTimeout( ATimeOut : Integer)
19707: Procedure SetTraceEvent( Event : TDBXTraceEvent)
19708: Procedure SetUserName( const UserName : string)
19709: Procedure SetWallpaper( Path : string);
19710: procedure ShellStyleClick(Sender: TObject);
19711: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
19712: Procedure ShowFileProperties( const FileName : string)
19713: Procedure ShowIncludelClick( Sender : TObject)
19714: Procedure ShowInterfaceslClick( Sender : TObject)
19715: Procedure ShowLastExceptionlClick( Sender : TObject)
19716: Procedure ShowMessage( const Msg : string)
19717: Procedure ShowMessageBig(const aText : string);
19718: Procedure ShowMessageBig2(const aText : string; aAutoSize: boolean);
19719: Procedure ShowMessageBig3(const aText : string; fSize: byte; aAutoSize: boolean);
19720: Procedure MsgBig(const aText : string); //alias
19721: procedure showMessage(mytext: string);
19722: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
19723: procedure ShowMessageFmt(const Msg: string; Params: array of const))
19724: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
19725: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
19726: Procedure ShowSearchDialog( const Directory : string)
19727: Procedure ShowSpecCharslClick( Sender : TObject)
19728: Procedure ShowBitmap(bmap: TBitmap); //draw in a form!
19729: Procedure ShredFile( const FileName : string; Times : Integer)
19730: procedure Shuffle(v0: TStringList);
19731: Procedure ShuffleList( var List : array of Integer; Count : Integer)
19732: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
19733: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
19734: Procedure SinCose( X : Extended; out Sin, Cos : Extended)
19735: Procedure Site( const ACommand : string)
19736: Procedure SkipEOL
19737: Procedure Sleep( ATIME : cardinal)
19738: Procedure Sleep( milliseconds : Cardinal)
19739: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
19740: Procedure SlinenumberslClick( Sender : TObject)
19741: Procedure Sort
19742: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
19743: procedure SoundAlarm; //beep seq
19744: procedure Speak(const sText: string) //async like voice
19745: procedure Speak2(const sText: string) //sync
19746: procedure Split(Str: string; SubStr: string; List: TStrings);
19747: Procedure SplitNameValuePair( const Line : string; var Name, Value : string)
19748: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
19749: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
19750: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
19751: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
19752: procedure SQLSyntaxlClick(Sender: TObject);
19753: Procedure SRand( Seed : RNG_IntType)
19754: Procedure Start
19755: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
19756: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
19757: //Ex. StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
19758: Procedure StartTransaction( TransDesc : TTransactionDesc)
19759: Procedure Status( var AStatusList : TStringList)
19760: Procedure StatusBar1DblClick( Sender : TObject)
19761: Procedure StepInto1Click( Sender : TObject)
19762: Procedure StepIt
19763: Procedure StepOut1Click( Sender : TObject)
19764: Procedure Stop
19765: procedure stopmp3;
19766: procedure StartWeb(aurl: string);
19767: Procedure StrToInt( integer; astr: string); //of system
19768: Procedure StrDispose( Str : PChar)
19769: procedure StrDispose(Str: PChar)
19770: Procedure StrReplace(var Str: String; Old, New: String);
19771: Procedure StretchDIBs( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
19772: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
19773: Procedure StringToBytes( Value : String; Bytes : array of byte)
19774: procedure StrSet(c : Char; I : Integer; var s : String);
19775: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19776: Procedure StructureMount( APath : String)
19777: procedure STYLECHANGED(SENDER:TOBJECT)
19778: Procedure Subselect( Node : TTreenode; Validate : Boolean)
19779: procedure Succ(X: int64);
19780: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
19781: procedure SwapChar(var X,Y: char); //swapX follows
19782: Procedure SwapFloats( var X, Y : Float)
19783: procedure SwapGrid(grd: TStringGrid);
19784: Procedure SwapOrd( var I, J : Byte);
19785: Procedure SwapOrd( var X, Y : Integer)
19786: Procedure SwapOrd1( var I, J : Shortint);
19787: Procedure SwapOrd2( var I, J : Smallint);
19788: Procedure SwapOrd3( var I, J : Word);
19789: Procedure SwapOrd4( var I, J : Integer);
19790: Procedure SwapOrd5( var I, J : Cardinal);

```

```

19791: Procedure SwapOrd6( var I, J : Int64);
19792: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
19793: Procedure Synchronize1( Method : TMethod);
19794: procedure SyntaxCheck1Click(Sender: TObject);
19795: Procedure SysFreeString(const S: WideString); stdcall;
19796: Procedure TakeOver( Other : TLinearBitmap)
19797: Procedure Talkln(const sText: string) //async voice
19798: procedure tbtn6resClick(Sender: TObject);
19799: Procedure tbtnUseCaseClick( Sender : TObject)
19800: procedure TerminalStyle1Click(Sender: TObject);
19801: Procedure Terminate
19802: Procedure texSyntax1Click( Sender : TObject)
19803: procedure TextOut(X, Y: Integer; Text: string);
19804: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
19805: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
19806: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
19807: Procedure TextStart
19808: procedure TILE
19809: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
19810: Procedure TitleClick( Column : TColumn)
19811: Procedure ToDo
19812: procedure Tone(500 hz, 10000 ms length);
19813: procedure toolbtnTutorialClick(Sender: TObject);
19814: Procedure Trace1( AURL : string; const AResponseContent : TStream);
19815: Procedure TransferMode( ATransferMode : TIIdFTPTTransferMode)
19816: Procedure Truncate
19817: procedure Tutorial101Click(Sender: TObject);
19818: procedure Tutorial10Statistics1Click(Sender: TObject);
19819: procedure Tutorial11Forms1Click(Sender: TObject);
19820: procedure Tutorial12SQL1Click(Sender: TObject);
19821: Procedure tutorial1Click( Sender : TObject)
19822: Procedure tutorial21Click( Sender : TObject)
19823: Procedure tutorial31Click( Sender : TObject)
19824: Procedure tutorial4Click( Sender : TObject)
19825: Procedure Tutorial5Click( Sender : TObject)
19826: procedure Tutorial6Click(Sender: TObject);
19827: procedure Tutorial91Click(Sender: TObject);
19828: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
19829: procedure UniqueString(var str: AnsiString)
19830: procedure UnloadLoadPackage(Module: HMODULE)
19831: Procedure Unlock
19832: Procedure UNMERGE( MENU : TMAINMENU)
19833: Procedure UnRegisterChanges( Value : TChangeLink)
19834: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
19835: Procedure UnregisterConversionType( const AType : TConvType)
19836: Procedure UnRegisterProvider( Prov : TCustomProvider)
19837: Procedure UPDATE
19838: Procedure UpdateBatch( AffectRecords : TAffectRecords)
19839: Procedure UPDATECURSORPOS
19840: Procedure UpdateFile
19841: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
19842: Procedure UpdateResponse( AResponse : TWebResponse)
19843: Procedure UpdateScrollBar
19844: Procedure UpdateView1Click( Sender : TObject)
19845: procedure UpdateExeResource(Const Source,Dest:string); //!
19846: procedure Val(const s: string; var n, z: Integer)
19847: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
19848: Procedure VarFMTBCDCreate( var ADest : Variant; const ABcd : TBcd);
19849: Procedure VariantAdd( const src : Variant; var dst : Variant)
19850: Procedure VariantAnd( const src : Variant; var dst : Variant)
19851: Procedure VariantArrayRedim( var V : Variant; High : Integer)
19852: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
19853: Procedure VariantClear( var V : Variant)
19854: Procedure VariantCpy( const src : Variant; var dst : Variant)
19855: Procedure VariantDiv( const src : Variant; var dst : Variant)
19856: Procedure VariantMod( const src : Variant; var dst : Variant)
19857: Procedure VariantMul( const src : Variant; var dst : Variant)
19858: Procedure VariantOr( const src : Variant; var dst : Variant)
19859: Procedure VariantPutElement( var V : Variant; const data : Variant; il : integer);
19860: Procedure VariantPutElement1( var V : Variant; const data : Variant; il, i2 : integer);
19861: Procedure VariantPutElement2( var V : Variant; const data : Variant; il, i2, i3 : integer);
19862: Procedure VariantPutElement3( var V : Variant; const data : Variant; il, i2, i3, i4 : integer);
19863: Procedure VariantPutElement4( var V : Variant; const data : Variant; il, i2, i3, i4, i5 : integer);
19864: Procedure VariantShl( const src : Variant; var dst : Variant)
19865: Procedure VariantShr( const src : Variant; var dst : Variant)
19866: Procedure VariantSub( const src : Variant; var dst : Variant)
19867: Procedure VariantXor( const src : Variant; var dst : Variant)
19868: Procedure VarCastError;
19869: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
19870: Procedure VarInvalidOp
19871: Procedure VarInvalidNullOp
19872: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
19873: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
19874: Procedure VarArrayCreateError
19875: Procedure VarResultCheck( AResult : HRESULT);
19876: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
19877: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
19878: Function VarTypeAsText( const AType : TVarType) : string
19879: procedure Voice(const sText: string) //async

```

```

19880: procedure Voice2(const sText: string) //sync
19881: Procedure WaitMiliSeconds( AMSec : word)
19882: Procedure WaitMS( AMSec : word');
19883: procedure WebCamPic(picname: string); //eg: c:\mypic.png
19884: Procedure WideAppend( var dst : WideString; const src : WideString)
19885: Procedure WideAssign( var dst : WideString; var src : WideString)
19886: Procedure WideDelete( var dst : WideString; index, count : Integer)
19887: Procedure WideFree( var s : WideString)
19888: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
19889: Procedure WideFromPChar( var dst : WideString; src : PChar)
19890: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
19891: Procedure WideSetLength( var dst : WideString; len : Integer)
19892: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
19893: Procedure WidestringToBytes( Value : WideString; Bytes : array of byte)
19894: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
19895: Procedure Wininet_HttpGet(const Url: string; Stream:TStream);
19896: Procedure HttpGet(const Url: string; Stream:TStream);
19897: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIddBytes; Index : integer)
19898: Procedure WordWrap1Click( Sender : TObject)
19899: Procedure Write( const AOut : string)
19900: Procedure Write( Socket : TSocket)
19901: procedure Write(S: string);
19902: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
19903: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
19904: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
19905: procedure WriteBuffer(Buffer:String;Count:LongInt)
19906: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
19907: Procedure WriteChar( AValue : Char)
19908: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
19909: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
19910: Procedure WriteFloat( const Section, Name : string; Value : Double)
19911: Procedure WriteHeader( AHeader : TStrings)
19912: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
19913: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
19914: Procedure Writeln( const AOut : string)
19915: procedure Writeln(s: string);
19916: Procedure WriteLog( const FileName, LogLine : string)
19917: Procedure WriteRFCReply( AReply : TIIdRFCReply)
19918: Procedure WriteRFCStrings( AStrings : TStrings)
19919: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
19920: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
19921: Procedure WriteString( const Section, Ident, Value : String)
19922: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
19923: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
19924: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
19925: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19926: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19927: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
19928: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
19929: procedure XMLSyntax1Click(Sender: TObject);
19930: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
19931: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
19932: Procedure ZeroFillStream( Stream : TMemoryStream)
19933: procedure XMLSyntax1Click(Sender: TObject);
19934: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
19935: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
19936: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
19937: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
19938: procedure(Sender, Source: TObject; X, Y: Integer)
19939: procedure(Sender, Target: TObject; X, Y: Integer)
19940: procedure(Sender: TObject; ASection, AWidth: Integer)
19941: procedure(Sender: TObject; ScrollCode: TScrollCode; var ScrollPos: Integer)
19942: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
19943: procedure(Sender: TObject; var Action: TCloseAction)
19944: procedure(Sender: TObject; var CanClose: Boolean)
19945: procedure(Sender: TObject; var Key: Char);
19946: ProcedureName ProcedureNames ProcedureParametersCursor @
19947:
19948: *****Now Constructors constructor *****
19949: Size is: 1248 1115 996 628 550 544 501 459 (381)
19950: Attach( VersionInfoData : Pointer; Size : Integer)
19951: constructor Create( ABuckets : TBucketListSizes)
19952: Create( ACallBackWnd : HWND)
19953: Create( AClient : TCustomTaskDialog)
19954: Create( AClient : TIIdTelnet)
19955: Create( ACollection : TCollection)
19956: Create( ACollection : TFavoriteLinkItems)
19957: Create( ACollection : TTaskDialogButtons)
19958: Create( AConnection : TIIdCustomHTTP)
19959: Create( ACreateSuspended : Boolean)
19960: Create( ADataSet : TCustomSQLDataSet)
19961: CREATE( ADATASET : TDATASET)
19962: Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
19963: Create( AGrid : TCustomDBGrid)
19964: Create( AGrid : TStringGrid; AIndex : Longint)
19965: Create( AHTTP : TIIdCustomHTTP)
19966: Create( AListItems : TlistItems)
19967: Create( AOnBytesRemoved : TIIdBufferBytesRemoved)
19968: Create( AOnBytesRemoved : TIIdBufferBytesRemoved)

```

```

19969: Create( AOwner : TCommonCalendar)
19970: Create( AOwner : TComponent)
19971: CREATE( AOWNER : TCOMPONENT)
19972: Create( AOwner : TCustomListView)
19973: Create( AOwner : TCustomOutline)
19974: Create( AOwner : TCustomRichEdit)
19975: Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
19976: Create( AOwner : TCustomTreeView)
19977: Create( AOwner : TIIdUserManager)
19978: Create( AOwner : TListItems)
19979: Create(AOwner:TObj;Handle:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
19980: CREATE( AOWNER : TPERSISTENT)
19981: Create( AOwner : TPersistent)
19982: Create( AOwner : TTable)
19983: Create( AOwner : TTreeNodes)
19984: Create( AOwner : TWinControl; const ClassName : string)
19985: Create( AParent : TIIdCustomHTTP)
19986: Create( AParent : TUpdateTree; AResolver : TCustomResolver)
19987: Create( AProvider : TBaseProvider)
19988: Create( AProvider : TCustomProvider);
19989: Create( AProvider : TDataSetProvider)
19990: Create( ASocket : TCustomWinSocket; TimeOut : Longint)
19991: Create( ASocket : TSocket)
19992: Create( AStrings : TWideStrings)
19993: Create( AToolBar : TToolBar)
19994: Create( ATreeNodes : TTreeNodes)
19995: Create( Autofill : boolean)
19996: Create( AWebPageInfo : TAbstractWebPageInfo)
19997: Create( AWebRequest : TWebRequest)
19998: Create( Collection : TCollection)
19999: Create( Collection : TIIdMessageParts; ABody : TStrings)
20000: Create( Collection : TIIdMessageParts; const AFileName : TFileName)
20001: Create( Column : TColumn)
20002: Create( const AConvFamily : TConvFamily; const ADescription : string)
20003: Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
20004: Create( const AConvFamily:TCConvFamily;const ADescription:string;const AToCommonProc,
AFromComProc:TConversionProc)
20005: Create( const AInitialState : Boolean; const AManualReset : Boolean)
20006: Create( const ATabSet : TTabSet)
20007: Create( const Compensate : Boolean)
20008: Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
20009: Create( const FileName : string)
20010: Create( const FileName : string; FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
: Int64; const SecAttr : PSecurityAttributes);
20011: Create( const FileName : string; FileMode : WordfmShareDenyWrite)
20012: Create( const MaskValue : string)
20013: Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
20014: Create( const Prefix : string)
20015: Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
20016: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
20017: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
20018: Create( CoolBar : TCoolBar)
20019: Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
20020: Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
20021: Create( DataSet : TDataSet; const
Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
DepFields : TBits; FieldMap : TFieldMap)
20022: Create( DBCtrlGrid : TDBCtrlGrid)
20023: Create( DSTableProducer : TDSTableProducer)
20024: Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
20025: Create( ErrorCode : DBIResult)
20026: Create( Field : TBlobField; Mode : TBlobStreamMode)
20027: Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
20028: Create( HeaderControl : TCustomHeaderControl)
20029: Create( HTTPRequest : TWebRequest)
20030: Create( iStart : integer; sText : string)
20031: Create( iValue : Integer)
20032: Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
20033: Create( MciErrNo : MCIERROR; const Msg : string)
20034: Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
20035: Create( Message : string; ErrorCode : DBResult)
20036: Create( Msg : string)
20037: Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
20038: Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
20039: Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
20040: Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
20041: Create(oSource:TniRegularExpressionState;oDestination:TniRegularExpression;xCharacts:TCharSet;bLambda:bool)
20042: Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
20043: Create( Owner : TCustomComboBoxEx)
20044: CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
20045: Create( Owner : TPersistent)
20046: Create( Params : TStrings) Create( Size : Cardinal)
20047: Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
20048: Create( StatusBar : TCustomStatusBar)
20049: Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
20050: Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
20051: Create(AHandle:Integer)
20052: Create(AOwner: TComponent); virtual;
20053: Create(const AURI : string)

```

```

20054: Create(FileName:String;Mode:Word)
20055: Create(Instance:THandle;ResName:String;ResType:PChar)
20056: Create(Stream : TStream)
20057: Create( ADataset : TDataset);
20058: Create( const FileHandle:THandle; const Name:string;Protect:Cardinal; const MaximumSize:Int64; const
SecAttr:PSecurityAttributes);
20059: Create( const FileName : string; const AIIndexOption : TJclMappedTextReaderIndex);
20060: Create2( Other : TObject);
20061: CreateAt(FileMap : TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
20062: CreateError(const anErrCode: Integer; const asReplyMessage: string; const asErrorMessage: string)
20063: CreateFmt( MciErrNo : MCIERROr; const Msg : string; const Args : array of const)
20064: CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
20065: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
20066: CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
20067: CreateRes( Ident : Integer);
20068: CreateRes( MciErrNo : MCIERROr; Ident : Integer)
20069: CreateRes( ResStringRec : PResStringRec);
20070: CreateResHelp( Ident : Integer; AHelpContext : Integer);
20071: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
20072: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
20073: CreateSize( AWidth, AHeight : Integer)
20074: Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
20075:
20076: -----
20077: unit UPSI_MathMax;
20078: -----
20079: CONSTS
20080: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
20081: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
20082: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
20083: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
20084: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
20085: CbrtPi: Float = 1.464591887561523630201425272638; // CubeRoot(Pi)
20086: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant
20087: PiJ: Float = 3.1415926535897932384626433832795; // PI
20088: PI: Extended = 3.1415926535897932384626433832795;
20089: PiOn2: Float = 1.5707963267948966192313216916398; // PI / 2
20090: PiOn3: Float = 1.0471975511965977461542144610932; // PI / 3
20091: PiOn4: Float = 0.78539816339744830961566084581988; // PI / 4
20092: Sqrt2: Float = 1.4142135623730950488016887242097; // Sqrt(2)
20093: Sqrt3: Float = 1.732050807568877293527463415059; // Sqrt(3)
20094: Sqrt5: Float = 2.2360679774997896964091736687313; // Sqrt(5)
20095: Sqrt10: Float = 3.1622776601683793319988935444327; // Sqrt(10)
20096: SqrtPi: Float = 1.7724538509055160272981674833411; // Sqrt(Pi)
20097: Sqrt2Pi: Float = 2.506628274631000502415765284811; // Sqrt(2 * PI)
20098: TwoPi: Float = 6.283185307179586476925286766559; // 2 * PI
20099: ThreePi: Float = 9.42447779607693797153879301498385; // 3 * PI
20100: Ln2: Float = 0.69314718055994530941723212145818; // Ln(2)
20101: Ln10: Float = 2.3025850929940456840179914546844; // Ln(10)
20102: LnPi: Float = 1.1447298858494001741434273513531; // Ln(Pi)
20103: Log2J: Float = 0.30102999566398119521373889472449; // Log10(2)
20104: Log3: Float = 0.47712125471966243729502790325512; // Log10(3)
20105: LogPi: Float = 0.4971498726941338543512682882909; // Log10(Pi)
20106: LogE: Float = 0.43429448190325182765112891891661; // Log10(E)
20107: E: Float = 2.7182818284590452353602874713527; // Natural constant
20108: hLn2Pi: Float = 0.9189385320467274178032973640562; // Ln(2*PI)/2
20109: inv2Pi: Float = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
20110: TwoToPower63: Float = 9223372036854775808.0; // 2^63
20111: GoldenMean: Float = 1.618033988749894848204586834365638; // GoldenMean
20112: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
20113: RadCor : Double = 57.29577951308232; {number of degrees in a radian}
20114: StDelta : Extended = 0.00001; {delta for difference equations}
20115: StEpsilon : Extended = 0.00001; {epsilon for difference equations}
20116: StMaxIterations : Integer = 100; {max attempts for convergence}
20117:
20118: procedure SIRegister_StdConvs(CL: TPPascalCompiler);
20119: begin
20120:   MetersPerInch = 0.0254; // [1]
20121:   MetersPerFoot = MetersPerInch * 12;
20122:   MetersPerYard = MetersPerFoot * 3;
20123:   MetersPerMile = MetersPerFoot * 5280;
20124:   MetersPerNauticalMiles = 1852;
20125:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
20126:   MetersPerLightSecond = 2.99792458E8; // [5]
20127:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
20128:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
20129:   MetersPerCubit = 0.4572; // [6][7]
20130:   MetersPerFathom = MetersPerFoot * 6;
20131:   MetersPerFurlong = MetersPerYard * 220;
20132:   MetersPerHand = MetersPerInch * 4;
20133:   MetersPerPace = MetersPerInch * 30;
20134:   MetersPerRod = MetersPerFoot * 16.5;
20135:   MetersPerChain = MetersPerRod * 4;
20136:   MetersPerLink = MetersPerChain / 100;
20137:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
20138:   MetersPerPica = MetersPerPoint * 12;
20139:
20140:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
20141:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;

```

```

20142:     SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
20143:     SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
20144:     SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
20145:     SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
20146:
20147:     CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
20148:     CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
20149:     CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
20150:     CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
20151:     CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
20152:     CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
20153:     CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
20154:     CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
20155:
20156:     CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
20157:     CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
20158:     CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
20159:     CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
20160:     CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
20161:     CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
20162:     CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
20163:     CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
20164:     CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
20165:     CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
20166:     CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
20167:     CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
20168:     CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
20169:     CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
20170:
20171:     CubicMetersPerUKGallon = 0.00454609; // [2][7]
20172:     CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
20173:     CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
20174:     CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
20175:     CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
20176:     CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
20177:     CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
20178:     CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
20179:     CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
20180:
20181:     GramsPerPound = 453.59237; // [1][7]
20182:     GramsPerDrams = GramsPerPound / 256;
20183:     GramsPerGrains = GramsPerPound / 7000;
20184:     GramsPerTons = GramsPerPound * 2000;
20185:     GramsPerLongTons = GramsPerPound * 2240;
20186:     GramsPerOunces = GramsPerPound / 16;
20187:     GramsPerStones = GramsPerPound * 14;
20188:
20189:     MaxAngle 9223372036854775808.0;
20190:     MaxTanH 5678.2617031470719747459655389854);
20191:     MaxFactorial( 1754);
20192:     MaxFloatingPoint(1.189731495357231765085759326628E+4932);
20193:     MinFloatingPoint(3.3621031431120935062626778173218E-4932);
20194:     MaxTanH( 354.89135644669199842162284618659);
20195:     MaxFactorial'LongInt'( 170);
20196:     MaxFloatingPointD(1.797693134862315907729305190789E+308);
20197:     MinFloatingPointD(2.2250738585072013830902327173324E-308);
20198:     MaxTanH( 44.361419555836499802702855773323);
20199:     MaxFactorial'LongInt'( 33);
20200:     MaxFloatingPoints( 3.4028236692093846346337460743177E+38);
20201:     MinFloatingPoints( 1.1754943508222875079687365372222E-38);
20202:     PiExt( 3.1415926535897932384626433832795);
20203:     RatioDegToRad( PiExt / 180.0);
20204:     RatioGradToRad( PiExt / 200.0);
20205:     RatioDegToGrad( 200.0 / 180.0);
20206:     RatioGradToDeg( 180.0 / 200.0);
20207:     Crc16PolynomCCITT'LongWord $1021);
20208:     Crc16PolynomIBM'LongWord $8005);
20209:     Crc16Bits'LongInt'( 16);
20210:     Crc16Bytes'LongInt'( 2);
20211:     Crc16HighBit'LongWord $8000);
20212:     NotCrc16HighBit','LongWord $7FFF);
20213:     Crc32PolynomIEEE','LongWord $04C11DB7);
20214:     Crc32PolynomCastagnoli','LongWord $1EDC6F41);
20215:     Crc32Koopman ','LongWord $741B8CD7);
20216:     Crc32Bits','LongInt'( 32);
20217:     Crc32Bytes','LongInt'( 4);
20218:     Crc32HighBit','LongWord $80000000);
20219:     NotCrc32HighBit','LongWord $7FFFFFFF);
20220:
20221:     MinByte      = Low(Byte);
20222:     MaxByte      = High(Byte);
20223:     MinWord      = Low(Word);
20224:     MaxWord      = High(Word);
20225:     MinShortInt  = Low(ShortInt);
20226:     MaxShortInt  = High(ShortInt);
20227:     MinSmallInt  = Low(SmallInt);
20228:     MaxSmallInt  = High(SmallInt);
20229:     MinLongWord   = LongWord(Low(LongWord));
2030:     MaxLongWord   = LongWord(High(LongWord));

```

```

20231: MinLongInt    = LongInt(Low(LongInt));
20232: MaxLongInt    = LongInt(High(LongInt));
20233: MinInt64      = Int64(Low(Int64));
20234: MaxInt64      = Int64(High(Int64));
20235: MinInteger    = Integer(Low(Integer));
20236: MaxInteger    = Integer(High(Integer));
20237: MinCardinal   = Cardinal(Low(Cardinal));
20238: MaxCardinal   = Cardinal(High(Cardinal));
20239: MinNativeUInt  = NativeUInt(Low(NativeUInt));
20240: MaxNativeUInt  = NativeUInt(High(NativeUInt));
20241: MinNativeInt   = NativeInt(Low(NativeInt));
20242: MaxNativeInt   = NativeInt(High(NativeInt));
20243: Function Cosh( const Z : Float) : Float;
20244: Function Sinh( const Z : Float) : Float;
20245: Function TanH( const Z : Float) : Float;
20246:
20247: //*****from DMath.Lib of types.inc in source\dmath_dll
20248: InvLn2          = 1.44269504088896340736; { 1/Ln(2) }
20249: InvLn10         = 0.43429448190325182765; { 1/Ln(10) }
20250: TwoPi          = 6.28318530717958647693; { 2*Pi }
20251: PiDiv2         = 1.57079632679489661923; { Pi/2 }
20252: SqrtPi         = 1.77245385090551602730; { Sqrt(Pi) }
20253: Sqrt2Pi        = 2.50662827463100050242; { Sqrt(2*Pi) }
20254: InvSqrt2Pi     = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
20255: LnSqrt2Pi      = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
20256: Ln2PiDiv2      = 0.91893853320467274178; { Ln(2*Pi)/2 }
20257: Sqrt2           = 1.41421356237309504880; { Sqrt(2) }
20258: Sqrt2Div2      = 0.7071067818654752440; { Sqrt(2)/2 }
20259: Gold           = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
20260: CGold          = 0.38196601125010515179; { 2 - GOLD }
20261: MachEp         = 2.220446049250313E-16; { 2^(-52) }
20262: MaxNum          = 1.797693134862315E+308; { 2^1024 }
20263: MinNum          = 2.225073858507202E-308; { 2^(-1022) }
20264: MaxLog          = 709.7827128933840;
20265: MinLog          = -708.3964185322641;
20266: MaxFac          = 170;
20267: MaxGam          = 171.624376956302;
20268: MaxLgm          = 2.556348E+305;
20269: SingleCompareDelta = 1.0E-34;
20270: DoubleCompareDelta = 1.0E-280;
20271: {$IFDEF CLR}
20272: ExtendedCompareDelta = DoubleCompareDelta;
20273: {$ELSE}
20274: ExtendedCompareDelta = 1.0E-4400;
20275: {$ENDIF}
20276: Bytes1KB         = 1024;
20277: Bytes1MB         = 1024 * Bytes1KB;
20278: Bytes1GB         = 1024 * Bytes1MB;
20279: Bytes64KB        = 64 * Bytes1KB;
20280: Bytes64MB        = 64 * Bytes1MB;
20281: Bytes2GB         = 2 * LongWord(Bytes1GB);
20282: clBlack32', $FF000000 );
20283: clDimGray32', $FF3F3F3F );
20284: clGray32', $FF7F7F7F );
20285: clLightGray32', $FFBFBFBF );
20286: clWhite32', $FFFFFF );
20287: clMaroon32', $FF7F0000 );
20288: clGreen32', $FF007F00 );
20289: clOlive32', $FF7F7F00 );
20290: clNavy32', $FF00007F );
20291: clPurple32', $FF7F007F );
20292: clTeal32', $FF007F7F );
20293: clRed32', $FFF00000 );
20294: clLime32', $FF00FF00 );
20295: clYellow32', $FFFFFF00 );
20296: clBlue32', $FF0000FF );
20297: clFuchsia32', $FFFF00FF );
20298: clAqua32', $FF00FFFF );
20299: clAliceBlue32', $FFF0F8FF );
20300: clAntiqueWhite32', $FFFAEBD7 );
20301: clAquamarine32', $FF7FFF04 );
20302: clAzure32', $FFF0FFFF );
20303: clBeige32', $FFF5F5DC );
20304: clBisque32', $FFFFE4C4 );
20305: clBlancheDalmond32', $FFFFEBBCD );
20306: clBlueViolet32', $FF8A2B2E );
20307: clBrown32', $FFA52A2A );
20308: clBurlyWood32', $FFDEB887 );
20309: clCadetblue32', $FF5F9EA0 );
20310: clChartreuse32', $FF7FFF00 );
20311: clChocolate32', $FFD2691E );
20312: clCoral32', $FFFF7F50 );
20313: clCornFlowerBlue32', $FF6495ED );
20314: clCornSilk32', $FFFFF8DC );
20315: clCrimson32', $FFDC143C );
20316: clDarkBlue32', $FF00008B );
20317: clDarkCyan32', $FF008B8B );
20318: clDarkGoldenRod32', $FFB8860B );
20319: clDarkGray32', $FFA9A9A9 );

```

```
20320:    clDarkGreen32', $FF006400 ));  
20321:    clDarkGrey32', $FFA9A9A9 ));  
20322:    clDarkKhaki32', $FFBDB76B ));  
20323:    clDarkMagenta32', $FF8B008B ));  
20324:    clDarkOliveGreen32', $FF556B2F ));  
20325:    clDarkOrange32', $FFFFF8C00 ));  
20326:    clDarkOrchid32', $FF9932CC ));  
20327:    clDarkRed32', $FF8B0000 ));  
20328:    clDarkSalmon32', $FFE9967A ));  
20329:    clDarkSeaGreen32', $FF8FB8CF ));  
20330:    clDarkSlateBlue32', $FF483D8B ));  
20331:    clDarkSlateGray32', $FF2F4F4F ));  
20332:    clDarkSlateGrey32', $FF2F4F4F ));  
20333:    clDarkTurquoise32', $FF00CED1 ));  
20334:    clDarkViolet32', $FF9400D3 ));  
20335:    clDeepPink32', $FFFF1493 ));  
20336:    clDeepSkyBlue32', $FF00BFFF ));  
20337:    clDodgerBlue32', $FF1E90FF ));  
20338:    clFireBrick32', $FFB22222 ));  
20339:    clFloralWhite32', $FFFFFFFA0 ));  
20340:    clGainsboro32', $FFDCDCDC ));  
20341:    clGhostWhite32', $FFF8F8FF ));  
20342:    clGold32', $FFFFD700 ));  
20343:    clGoldenRod32', $FFDA520 ));  
20344:    clGreenYellow32', $FFADFF2F ));  
20345:    clGrey32', $FF808080 ));  
20346:    clHoneyDew32', $FFF0FF0 ));  
20347:    clHotPink32', $FFFF69B4 ));  
20348:    clIndianRed32', $FFCD5C5C ));  
20349:    clIndigo32', $FF4B0082 ));  
20350:    clIvory32', $FFFFFFF0 ));  
20351:    clKhaki32', $FFFOE68C ));  
20352:    clLavender32', $FFE6E6FA ));  
20353:    clLavenderBlush32', $FFFFF0F5 ));  
20354:    clLawnGreen32', $FF7FCFC00 ));  
20355:    clLemonChiffon32', $FFFFFFACD ));  
20356:    clLightBlue32', $FFADD8E6 ));  
20357:    clLightCoral32', $FFF08080 ));  
20358:    clLightCyan32', $FFE0FFF ));  
20359:    clLightGoldenRodYellow32', $FFFAFAD2 ));  
20360:    clLightGreen32', $FF90EE90 ));  
20361:    clLightGrey32', $FFD3D3D3 ));  
20362:    clLightPink32', $FFFFB6C1 ));  
20363:    clLightSalmon32', $FFFA07A ));  
20364:    clLightSeagreen32', $FF20B2AA ));  
20365:    clLightSkyblue32', $FF87CEFA ));  
20366:    clLightSlategray32', $FF778899 ));  
20367:    clLightSlategrey32', $FF778899 ));  
20368:    clLightSteelblue32', $FFB0C4DE ));  
20369:    clLightYellow32', $FFFFFFE0 ));  
20370:    clLtGray32', $FFC0C0C0 ));  
20371:    clMedGray32', $FFA0A0A4 ));  
20372:    clDkGray32', $FF808080 ));  
20373:    clMoneyGreen32', $FFC0DCC0 ));  
20374:    clLegacySkyBlue32', $FFA6CAF0 ));  
20375:    clCream32', $FFFFFFBF0 ));  
20376:    clLimeGreen32', $FF32CD32 ));  
20377:    clLinen32', $FFFAF0E6 ));  
20378:    clMediumAquamarine32', $FF66CDAA ));  
20379:    clMediumBlue32', $FF0000CD ));  
20380:    clMediumOrchid32', $FFBA55D3 ));  
20381:    clMediumPurple32', $FF9370DB ));  
20382:    clMediumSeaGreen32', $FF3CB371 ));  
20383:    clMediumSlateBlue32', $FF7B68E9 ));  
20384:    clMediumSpringGreen32', $FF00FA9A ));  
20385:    clMediumTurquoise32', $FF48D1CC ));  
20386:    clMediumVioletRed32', $FFC71585 ));  
20387:    clMidnightBlue32', $FF191970 ));  
20388:    clMintCream32', $FFF5FFA ));  
20389:    clMistyRose32', $FFFFFFE4E1 ));  
20390:    clMoccasin32', $FFFFFFE4B5 ));  
20391:    clNavajoWhite32', $FFFFFFDEAD ));  
20392:    clOldLace32', $FFFDF5E6 ));  
20393:    clOliveDrab32', $FFGB8E23 ));  
20394:    clOrange32', $FFFFFFA500 ));  
20395:    clOrangeRed32', $FFFFFF4500 ));  
20396:    clOrchid32', $FFDA70D6 ));  
20397:    clPaleGoldenRod32', $FFEEE8AA ));  
20398:    clPaleGreen32', $FF98FB98 ));  
20399:    clPaleTurquoise32', $FFAFEEEE ));  
20400:    clPaleVioletred32', $FFDB7093 ));  
20401:    clPapayaWhip32', $FFFFFFFD5 ));  
20402:    clPeachPuff32', $FFFFFFDAB9 ));  
20403:    clPeru32', $FFCD853F ));  
20404:    clPlum32', $FFDDA0DD ));  
20405:    clPowderBlue32', $FFB0E0E6 ));  
20406:    clRosyBrown32', $FFB8C8F8F ));  
20407:    clRoyalBlue32', $FF4169E1 ));  
20408:    clSaddleBrown32', $FF8B4513 ));
```

```

20409:   clSalmon32', $FFFA8072 ));
20410:   clSandyBrown32', $FFF4A460 ));
20411:   clSeaGreen32', $FF2E8B57 ));
20412:   clSeaShell32', $FFFFFF5EE ));
20413:   clSienna32', $FFA0522D ));
20414:   clSilver32', $FFC0C0C0 ));
20415:   clSkyblue32', $FF87CEEB ));
20416:   clSlateBlue32', $FF6A5ACD ));
20417:   clSlateGrey32', $FF708090 ));
20418:   clSlateGrey32', $FF708090 ));
20419:   clSnow32', $FFFFFFAFA ));
```

```

20420:   clSpringgreen32', $FF00FF7F ));
20421:   clSteelblue32', $FF4682B4 ));
20422:   clTan32', $FFD2B48C ));
20423:   clThistle32', $FFD8BF08 ));
20424:   clTomato32', $FFFF6347 ));
20425:   clTurquoise32', $FF40E0D0 ));
20426:   clViolet32', $FFEE82EE ));
20427:   clWheat32', $FFF5DEB3 ));
20428:   clWhitesmoke32', $FFF5F5F5 ));
20429:   clYellowgreen32', $FF9ACD32 ));
20430:   clTrWhite32', $7FFFFFFF ));
20431:   clTrBlack32', $7F000000 ));
20432:   clTrRed32', $7FFF0000 ));
20433:   clTrGreen32', $7F00FF00 ));
20434:   clTrBlue32', $7F0000FF ));
20435: // Fixed point math constants
20436: FixedOne = $10000; FixedHalf = $7FFF;
20437: FixedPI = Round(PI * FixedOne);
20438: FixedToFloat = 1/FixedOne;
20439:
20440: Special Types
20441: ****
20442: type Complex = record //for complex numbers
20443:   X, Y : Float;
20444: end;
20445: type TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R : '
20446:   + Float; Theta : Float; end');
20447: type TVector      = array of Float;
20448: TIntVector    = array of Integer;
20449: TCompVector   = array of Complex;
20450: TBoolVector   = array of Boolean;
20451: TStringVector = array of String;
20452: TMatrix       = array of TVector;
20453: TIntMatrix    = array of TIntVector;
20454: TCompMatrix   = array of TCompVector;
20455: TBoolMatrix   = array of TBoolVector;
20456: TStringMatrix = array of TString;
20457: TByteArray    = array[0..32767] of byte; !
20458: THexArray     = array [0..15] of Char; // = '0123456789ABCDEF';
20459: TBitmapStyle  = (bsNormal, bsCentered, bsStretched);
20460: T2StringArray  = array of array of string;
20461: T2IntegerArray = array of array of integer;
20462: AddTypes('INT_PTR', 'Integer
20463: AddTypes('LONG_PTR', 'Integer
20464: AddTypes('UINT_PTR', 'Cardinal
20465: AddTypeS('ULONG_PTR', 'Cardinal
20466: AddTypes('DWORD_PTR', 'ULONG_PTR
20467: TIntegerDynArray', 'array of Integer
20468: TCardinalDynArray', 'array of Cardinal
20469: TWordDynArray', 'array of Word
20470: TSmallIntDynArray', 'array of SmallInt
20471: TByteDynArray', 'array of Byte
20472: TShortIntDynArray', 'array of ShortInt
20473: TInt64DynArray', 'array of Int64
20474: TLongWordDynArray', 'array of LongWord
20475: TSingleDynArray', 'array of Single
20476: TDoubleDynArray', 'array of Double
20477: TBooleanDynArray', 'array of Boolean
20478: TStringDynArray', 'array of string
20479: TWideStringDynArray', 'array of WideString
20480: TDynByteArray   = array of Byte;
20481: TDynShortintArray = array of Shortint;
20482: TDynSmallintArray = array of Smallint;
20483: TDynWordArray    = array of Word;
20484: TDynIntegerArray = array of Integer;
20485: TDynLongintArray = array of Longint;
20486: TDynCardinalArray = array of Cardinal;
20487: TDynInt64Array   = array of Int64;
20488: TDynExtendedArray = array of Extended;
20489: TDynDoubleArray  = array of Double;
20490: TDynSingleArray   = array of Single;
20491: TDynFloatArray   = array of Float;
20492: TDynPointerArray = array of Pointer;
20493: TDynStringArray   = array of string;
20494: TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
20495:   ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
20496: TSynSearchOptions = set of TSynSearchOption;
20497: TFloat = single

```

```

20498:   Float = double
20499:
20500:
20501: /* Project: IFSI_WinFormlpuzzle.pas BaseInclude RunTimeLib for maxbox *: pas_includebox.inc
20502: -----
20503: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
20504: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
20505: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
20506: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
20507: function CheckStringSum(vstring: string): integer;
20508: function HexToInt(HexNum: string): LongInt;
20509: function IntToBin(Int: Integer): String;
20510: function BinToInt(Binary: String): Integer;
20511: function HexToBin(HexNum: string): string; external2
20512: function BinToHex(Binary: String): string;
20513: function IntToFloat(i: Integer): double;
20514: function AddThousandSeparator(S: string; myChr: Char): string;
20515: function Max3(const X,Y,Z: Integer): Integer;
20516: procedure Swap(var X,Y: char); // faster without inline
20517: procedure ReverseString(var S: String);
20518: function CharToHexStr(Value: Char): string;
20519: function CharToUniCode(Value: Char): string;
20520: function Hex2Dec(Value: Str002): Byte;
20521: function HexStrCodeToStr(Value: string): string;
20522: function HexToStr(i: integer; value: string): string;
20523: function UniCodeToStr(Value: string): string;
20524: function CRC16(statement: string): string;
20525: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
20526: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
20527: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
20528: Procedure ExecuteCommand(executeFile, paramstring: string);
20529: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
20530: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
20531: procedure SearchAndOpenDoc(vfilenamepath: string);
20532: procedure ShowInterfaces(myFile: string);
20533: function Fact2(av: integer): extended;
20534: Function BoolToStr(B: Boolean): string;
20535: Function GCD(x, y : LongInt) : LongInt;
20536: function LCM(m,n: longint): longint;
20537: function GetASCII: string;
20538: function GetItemHeight(Font: TFont): Integer;
20539: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
20540: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
20541: function getHINSTANCE: longword;
20542: function getHMODULE: longword;
20543: function GetASCII: string;
20544: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
20545: function WordIsOk(const AWord: string; var VW: Word): boolean;
20546: function TwentyFourBitValueisOk(const AValue: string; var VI: Integer): boolean;
20547: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
20548: function SafeStr(const s: string): string;
20549: function ExtractUrlPath(const FileName: string): string;
20550: function ExtractUrlName(const FileName: string): string;
20551: function IsInternet: boolean;
20552: function RotateLeft1Bit_u32( Value: uint32): uint32;
20553: procedure LinearRegression(const KnownY:array of Double;const KnownX:array of Double;NData:Int;var LF:TStLinEst; ErrorStats: Bool);
20554: procedure getEnvironmentInfo;
20555: procedure AntiFreeze;
20556: function GetCPUSpeed: Double;
20557: function IsVirtualPcGuest : Boolean;
20558: function IsVmWareGuest : Boolean;
20559: procedure StartSerialDialog;
20560: function IsWoW64: boolean;
20561: function IsWow64String(var s: string): Boolean;
20562: procedure StartThreadDemo;
20563: Function RGB(R,G,B: Byte): TColor;
20564: Function SendIn(amess: string): boolean;
20565: Procedure maxbox;
20566: Function AspectRatio(aWidth, aHeight: Integer): String;
20567: function wget(aURL, afile: string): boolean;
20568: procedure PrintList(Value: TStringList);
20569: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
20570: procedure getEnvironmentInfo;
20571: procedure AntiFreeze;
20572: function getBitmap(apath: string): TBitmap;
20573: procedure ShowMessageBig(const aText : string);
20574: function YesNoDialog(const ACaption, AMsg: string): boolean;
20575: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
20576: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
20577: //function myStrToBytes(const Value: String): TBytes;
20578: //function myBytesToStr(const Value: TBytes): String;
20579: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20580: function getBitmap(apath: string): TBitmap;
20581: procedure ShowMessageBig(const aText : string);
20582: Function StrToBytes(const Value: String): TBytes;
20583: Function BytesToStr(const Value: TBytes): String;
20584: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20585: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retries:Int;var HostName:String):Bool;

```

```

20586: function FindInPaths(const fileName, paths : String) : String;
20587: procedure initHexArray(var hexn: THexArray);
20588: function josephusG(n,k: integer; var graphout: string): integer;
20589: function isPowerOf2(num: int64): boolean;
20590: function powerOf2(exponent: integer): int64;
20591: function getBigPI: string;
20592: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
20593: function GetASCIIline: string;
20594: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
20595:                                     pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
20596: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
20597: procedure AddComplexSoundObjectToList(newf,newp,newa,neww:integer; freqlist: TStrings);
20598: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
20599: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
20600: function isKeypressed: boolean;
20601: function Keypress: boolean;
20602: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
20603: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
20604: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
20605: function GetOSName: string;
20606: function GetOSVersion: string;
20607: function GetOSNumber: string;
20608: function getEnvironmentString: string;
20609: procedure StrReplace(var Str: String; Old, New: String);
20610: procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
20611: function getTeamViewerID: string;
20612: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
20613: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
20614: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
20615: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
20616: function StartSocketService: Boolean;
20617: procedure StartSocketServiceForm;
20618: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
20619: function GetFileList1(apath: string): TStringlist;
20620: procedure LetfileList(FileList: TStringlist; apath: string);
20621: procedure StartWeb(aurl: string);
20622: function GetTodayFiles(startdir, amask: string): TStringlist;
20623: function PortTCPISOpen(dwPort : Word; ipAddressStr: String): boolean;
20624: function JavahashCode(val: string): Integer;
20625: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
20626: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
20627: Procedure HideWindowForSeconds(secs: integer); //3 seconds
20628: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); //3 seconds
20629: Procedure ConvertToGray(Cnv: TCanvas);
20630: function GetFileDate(aFile:string; aWithTime:Boolean):string;
20631: procedure ShowMemory;
20632: function ShowMemory2: string;
20633: function getHostIP: string;
20634: procedure ShowBitmap(bmap: TBitmap);
20635: function GetOsVersionInfo: TOSVersionInfo; //thx to wischnewski
20636: function CreateDBGridForm(dblist: TStringList): TListBox;
20637: function isService: boolean;
20638: function isApplication: boolean;
20639: function isTerminalSession: boolean;
20640: function SetPrivilege(privilegeName: string; enable: boolean): boolean;
20641: procedure getScriptandRunAsk;
20642: procedure getScriptandRun(ascript: string);
20643: function VersionCheckAct: string;
20644: procedure getBox(aurl, extension: string);
20645: function CheckBox: string;
20646: function isNTFS: boolean;
20647: //procedure doWebCamPic;
20648: procedure doWebCamPic(picname: string);
20649: function readm: string;
20650: procedure getGEOMapandRunAsk;
20651: function GetMapX(C_form,apath: string; const Data: string): boolean;
20652: procedure GetGEOMap(C_form,apath: string; const Data: string);
20653: function GetMapXGeoReverse(C_form: string; const lat,long: string): string;
20654: //function RoundTo(const AValue: Extended;
20655: //                           const ADigit: TRoundToEXRangeExtended): Extended;
20656: function GetGeocodeCoord(C_form: string; const data:string; atxt: boolean): string;
20657: function GetGeoCoord(C_form: string; const data:string; atxt: boolean): string;
20658:   ex.: writeln(GetGeoCoord('xml','church cefalu sicily',true));
20659: function DownloadFile(SourceFile, DestFile: string): Boolean;
20660: function DownloadFileOpen(SourceFile, DestFile: string): Boolean;
20661: function OpenMap(const Data: string): boolean;
20662: function GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
20663: Function getFileCount(amask: string): integer;
20664: function CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TNavPos): string;
20665: procedure DebugLn(DebugLOGFILE: string; E: string);
20666: function IntToFloat(i: Integer): double;
20667: function AddThousandSeparator(S: string; myChr: Char): string;
20668: function mymcSendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
20669: function RoundTime(ADate: string; Rounding: Integer; bRound: Boolean): string;
20670: function DynamicDllCallName(Dll: String; const Name: String; HasResult: Boolean; var Returned: Cardinal;
20671: const Parameters: array of integer): Boolean; '';
20671: function DynamicDllCall(Dll: String; const Name: String; HasResult: Boolean; var Returned: Cardinal;
20671: const Parameters: array of integer): Boolean; '';
20672:

```

```

20673: End C:\maxbook\maxbox3\mX3999\maxbox3\source\IFSI_WinForm1puzzle.pas File loaded
20674:
20675: // News of 3.9.8 up
20676: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20677: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20678: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20679: JvChart - TJvChart Component - 2009 Public
20680: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
20681: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
20682: TAdoQuery.SQL.Add() fixed, ShlwAPI extensions, Indy HTTPHeader Extensions
20683: DMath DLL included incl. Demos
20684: Interface Navigator menu/View/Intf Navigator
20685: Unit Explorer menu/Debug/Units Explorer
20686: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
20687: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
20688: Script History to 9 Files WebServer light /Options/Addons/WebServer
20689: Full Text Finder, JVSSimLogic Simulator Package
20690: Halt-Stop Program in Menu, WebServer2, Stop Event ,
20691: Conversion Routines, Prebuild Forms, CodeSearch
20692: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20693: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20694: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20695: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLLoader, TJvPaintFX
20696: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
20697: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
20698: IDE Reflection API, Session Service Shell S3
20699: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
20700: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
20701: arduino map() function, PMRandom Generator
20702: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
20703: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
20704: REST Test Lib, Multilang Component, Forth Interpreter
20705: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
20706: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
20707: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20708: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20709: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
20710: QRCode Service, add more CFunctions like CDateTime of Synapse
20711: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
20712: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
20713: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
20714: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
20715: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
20716: BOLD Package, Indy Package5, maTRIX. MATHMAX
20717: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
20718: emax layers: system-package-component-unit-class-function-block
20719: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms
20720: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
20721: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
20722: OpenGL Game Demo: ..Options/Add Ons/Reversi
20723: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
20724: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
20725: 7% performance gain (hot spot profiling)
20726: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
20727: add 42 + 22 (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
20728: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
20729: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
20730:
20731: add routines in 3.9.7.5
20732: 097: procedure RIRegister_BarCodeScanner_Routines(S: TPSEExec);
20733: 996: procedure RIRegister_DBCtrls_Routines(S: TPSEExec);
20734: 069: procedure RIRegister_IdStrings_Routines(S: TPSEExec);
20735: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSEExec);
20736: 215: procedure RIRegister_PNGLoader_Routines(S: TPSEExec);
20737: 374: procedure RIRegister_SerDlgs_Routines(S: TPSEExec);
20738: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSEExec);
20739:
20740: ////////////////////////////// TestUnits //////////////////////////////
20741: SelftestPEM;
20742: SelfTestCFundamentUtils;
20743: SelfTestCFileUtils;
20744: SelfTestCDateTime;
20745: SelfTestCTimer;
20746: SelfTestCRandom;
20747: Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter'
20748:           Assert(WinPathToUnixPath('\c\d\f') = '/c/d/f', 'WinPathToUnixPath'
20749:
20750: // Note: There's no need for installing a client certificate in the
20751: // webbrowser. The server asks the webbrowser to send a certificate but
20752: // if nothing is installed the software will work because the server
20753: // doesn't check to see if a client certificate was supplied. If you want you can install:
20754: // file: c_cacert.p12 password: c_cakey
20755:
20756: TGraphicControl = class(TControl)
20757: private
20758:   FCanvas: TCanvas;
20759:   procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20760:   protected
20761:     procedure Paint; virtual;

```

```

20762:     property Canvas: TCanvas read FCanvas;
20763:   public
20764:     constructor Create(AOwner: TComponent); override;
20765:     destructor Destroy; override;
20766:   end;
20767:
20768:   TCustomControl = class(TWinControl)
20769:   private
20770:     FCanvas: TCanvas;
20771:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20772:   protected
20773:     procedure Paint; virtual;
20774:     procedure PaintWindow(DC: HDC); override;
20775:     property Canvas: TCanvas read FCanvas;
20776:   public
20777:     constructor Create(AOwner: TComponent); override;
20778:     destructor Destroy; override;
20779:   end;
20780:   RegisterPublishedProperties;
20781:   ('ONCHANGE', 'TNotifyEvent', iptrw);
20782:   ('ONCLICK', 'TNotifyEvent', iptrw);
20783:   ('ONDBLCLICK', 'TNotifyEvent', iptrw);
20784:   ('ONENTER', 'TNotifyEvent', iptrw);
20785:   ('ONEXIT', 'TNotifyEvent', iptrw);
20786:   ('ONKEYDOWN', 'TKeyEvent', iptrw);
20787:   ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
20788:   ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
20789:   ('ONMOUSEMOVE', 'TMouseEvent', iptrw);
20790:   ('ONMOUSEUP', 'TMouseEvent', iptrw);
20791: //*****
20792: // To stop the while loop, click on Options>Show Include (boolean switch)! Control a loop in a script with a form event:
20793: IncludeON; //control the while loop
20794: while maxform1.ShowInclude1.checked do begin //menu event Options>Show Include
20795: repeat {for it:= 1 to n do} until is keypressed //keypress in output window below (memo2)
20797:
20798: //-----
20799: //*****mX4 ini-file Configuration*****
20800: //-----
20801: using config file maxboxdef.ini           menu/Help/Config File
20802:
20803: //*** Definitions for maXbox mX3 ***
20804: [FORM]
20805: LAST_FILE=E:\maxbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
20806: FONTSIZE=14
20807: EXTENSION=txt
20808: SCREENX=1386
20809: SCREENY=1077
20810: MEMHEIGHT=350
20811: PRINTFONT=Courier New //GUI Settings
20812: LINENUMBERS=Y //line numbers at gutter in editor at left side
20813: EXCEPTIONLOG=Y //store excepts + success in 2 log files see below! -menu Debug>Show Last Exceptions
20814: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
20815: BOOTSCRIPT=Y //enabling load a boot script
20816: MEMORYREPORT=Y //shows memory report on closing maXbox
20817: MACRO=Y //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
20818: NAVIGATOR=N //shows function list at the right side of editor
20819: NAVWIDTH=350 //width of the right side interface list <CTRL L> >=200
20820: AUTOBOOKMARK=Y //sets on all functions a bookmark to jump
20821: [WEB]
20822: IPPORT=8080 //for internal webserver - menu /Options/Add Ons/WebServer
20823: IPHOST=192.168.1.53 //run as Administrator!
20824: ROOTCERT=filepathY
20825: SCERT=filepathY
20826: RSAKEY=filepathY
20827: VERSIONCHECK=Y
20828: APP=C:\WINDOWS\System32\calc.exe //set path to an external app
20829: MYSCRIPT=E:\maxbox3\mXGit39991\maxbox3\examples\330_myclock.txt //start script of menu /View/MyScript
20830:
20831: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
20832: Also possible to set report memory in script to override ini setting
20833: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
20834:
20835: After Change the ini file you can reload the file with ..../Help/Config Update
20836: //-----
20837: //*****mX4 maildef.ini ini-file Configuration*****
20838: //-----
20839: //*** Definitions for maXMail ***
20840: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
20841: [MAXMAIL]
20842: HOST=mailto:software@schule.ch
20843: USER=mailusername
20844: PASS=password
20845: PORT=10
20846: SSL=Y
20847: BODY=Y
20848: LAST=5
20849:
20850: ADO Connection String:

```

```

20851: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
20852: \452_dbtreeview2access.txt \452_dbtrv3accessUML2.txt
20853: program TestDbTreeViewMainForm2_ACCESS;
20854:   ConnectionString='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
20855:     +Exepath+'\examples\detail.mdb;Persist Security Info=False';
20856:   'Provider=MSDASQL.1;Persist Security Info=False;Extended
Properties="DSN=FB_EMPLOYEE;Driver=Firebird/InterBase(r)
driver;Dbname=C:\maxbox\maxbox3\mX3999\maxbox3\examples\EMPLOYEE.FDB;CHARSET=NONE;UID=SYSDBA;Role=Admin;"'
20857:
20858: OpenSSL Lib: unit ssl_openssl_lib;
20859: {$IFDEF CIL}
20860: const
20861: {$IFDEF LINUX}
20862: DLLSSLName = 'libssl.so';
20863: DLLUtilName = 'libcrypto.so';
20864: {$ELSE}
20865: DLLSSLName = 'ssleay32.dll';
20866: DLLUtilName = 'libeay32.dll';
20867: {$ENDIF}
20868: {$ELSE}
20869: var
20870: {$IFNDEF MSWINDOWS}
20871: {$IFDEF DARWIN}
20872: DLLSSLName: string = 'libssl.dylib';
20873: DLLUtilName: string = 'libcrypto.dylib';
20874: {$ELSE}
20875: DLLSSLName: string = 'libssl.so';
20876: DLLUtilName: string = 'libcrypto.so';
20877: {$ENDIF}
20878: {$ELSE}
20879: DLLSSLName: string = 'ssleay32.dll';
20880: DLLSSLName2: string = 'libssl32.dll';
20881: DLLUtilName: string = 'libeay32.dll';
20882: {$ENDIF}
20883: {$ENDIF}
20884:
20885: //-----
20886: //*****mX4 Macro Tags *****
20887: //-----
20888:
20889: asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
E:\maxbox\maxbox3\docs\maxbox_extract_funcList399.txt end
20890:
20891: //Tag Macros in ini-file configure
20892:
20893: asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
20894:
20895: //Tag Macros
20896: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
20897: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
20898: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
20899: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
20900: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
20901: 10199: SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
20902: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
20903: 10194: SearchAndCopy(memo1.lines, '#perf', perfTime, 11);
20904: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%.2f %.2f %.2f',
20905: [getUserNameWin, getComputernameWin, datetimetoStr(now),
20906: 10196: SearchAndCopy(memo1.lines, '#head', Format('%.2f %.2f %.2f',
20907: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename],11));
20908: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename],11);
20909: 10198: SearchAndCopy(memo1.lines, '#tech',Format('%.2f %.2f %.2f',
20910: [perfTime, numprocesssthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]],11);
20911: 10298: SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
20912: [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
20913:
20914: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
20915:
20916: //Replace Macros
20917: SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
20918: SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
20919: SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
20920: SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
20921: SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
20922: SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+Source', 8);
20923: ref: netcologne.dl.sourceforge.net/project/maxbox/maxbox3.zip
20924: SearchAndCopy(memo1.lines, '#tech'perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
20925: [perfTime,numprocesssthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]], 11);
20926: //#tech!84perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
20927: SearchAndCopy(memo1.lines, 'maxbox_extract_funcList399.txt
20928:
20929: //-----
20930: //*****mX4 ToDo List Tags ..../Help/ToDo List*****
20931: //-----
20932:
20933: while I < sl.Count do begin
20934: // if MatchesMask(sl[I], '*/? TODO ([a-zA-Z_]*#[1-9#])*:*') then
20935: if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
20936:   BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO

```

```

20937:     else if MatchesMask(sl[I], '*/? DONE (?#?#)*:*') then
20938:       BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
20939:     else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
20940:       BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
20941:     else if MatchesMask(sl[I], '*/? DONE(#?#)*:*') then
20942:       BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
20943:     else if MatchesMask(sl[I], '*/?TODO*:*') then
20944:       BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
20945:     else if MatchesMask(sl[I], '*/?DONE*:*') then
20946:       BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
20947:     Inc(I);
20948:   end;
20949:
20950: //-----
20951: //*****mX4 Public Tools API *****
20952: //-----
20953: file : unit uPSI_fMain.pas;           {$SOTAP} Open Tools API Catalog
20954: // Those functions concern the editor and preprocessor, all of the IDE
20955: Example: Call it with maxform1.InfoClick(self)
20956: Note: Call all Methods with maxForm1., e.g.:
20957:         maxForm1.ShellStyle1Click(self);
20958:
20959: procedure SIRegister_fMain(CL: TPSPascalCompiler);
20960: begin
20961:   Const ('BYTECODE','String 'bytecode.txt'
20962:   Const ('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
20963:   Const ('PSMODEL','String PS Modelfiles (*.uc)|*.UC
20964:   Const ('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
20965:   Const ('PSINC','String PS Includes (*.inc)|*.INC
20966:   Const ('DEFFILENAME','String 'firstdemo.txt
20967:   Const ('DEFINIFILE','String 'maxboxdef.ini
20968:   Const ('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
20969:   Const ('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
20970:   Const ('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
20971:   Const ('ALLOBJECTSLIST','String 'docs\VCL.pdf
20972:   Const ('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
20973:   Const ('ALLUNITLIST','String 'docs\maxbox3_9.xml');
20974:   Const ('INCLUDEBOX','String 'pas_includebox.inc
20975:   Const ('BOOTSCRIPT','String 'maxbootscript.txt
20976:   Const ('MBVERSION','String '3.9.9.190
20977:   Const ('VERSION','String '3.9.9.190
20978:   Const ('MBVER','String '399
20979:   Const ('MBVER1','Integer'(399);
20980:   Const ('MBVERIALL','Integer'(399190);
20981:   Const ('EXENAME','String 'maxbox3.exe
20982:   Const ('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
20983:   Const ('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
20984:   Const ('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
20985:   Const ('MXINTERNETCHECK','String 'www.ask.com
20986:   Const ('MXMAIL','String 'max@kleiner.com
20987:   Const ('TAB','Char #'09);
20988:   Const ('CODECOMPLETION','String 'bds_delphi.dci
20989:   SIRegister_TMaxForm1(CL);
20990: end;
20991:
20992: with FindClass('TForm'),'TMaxForm1') do begin
20993:   memo2', 'TMemo', iptrw);
20994:   memo1', 'TSynMemo', iptrw);
20995:   CB1SCList', 'TComboBox', iptrw);
20996:   mxNavigator', 'TComboBox', iptrw);
20997:   IPHost', 'string', iptrw);
20998:   IPPort', 'integer', iptrw);
20999:   COMPort', 'integer', iptrw); //3.9.6.4
21000:   Splitter1', 'TSplitter', iptrw);
21001:   PSScript', 'TPSScript', iptrw);
21002:   PS3D11Plugin', 'TPSD11Plugin', iptrw);
21003:   MainMenul', 'TMainMenu', iptrw);
21004:   Program1', 'TMenuItem', iptrw);
21005:   Compile1', 'TMenuItem', iptrw);
21006:   Files1', 'TMenuItem', iptrw);
21007:   open1', 'TMenuItem', iptrw);
21008:   Save2', 'TMenuItem', iptrw);
21009:   Options1', 'TMenuItem', iptrw);
21010:   Savebefore1', 'TMenuItem', iptrw);
21011:   Largefont1', 'TMenuItem', iptrw);
21012:   sBytecode1', 'TMenuItem', iptrw);
21013:   Saveas3', 'TMenuItem', iptrw);
21014:   Clear1', 'TMenuItem', iptrw);
21015:   Slinenumbers1', 'TMenuItem', iptrw);
21016:   About1', 'TMenuItem', iptrw);
21017:   Search1', 'TMenuItem', iptrw);
21018:   SynPasSyn1', 'TSynPasSyn', iptrw);
21019:   memo1', 'TSynMemo', iptrw);
21020:   SynEditSearch1', 'TSynEditSearch', iptrw);
21021:   WordWrap1', 'TMenuItem', iptrw);
21022:   XPManifest1', 'TXPManifest', iptrw);
21023:   SearchNext1', 'TMenuItem', iptrw);
21024:   Replace1', 'TMenuItem', iptrw);
21025:   PSImport_Controls1', 'TPSImport_Controls', iptrw);

```

```

21026: PSImport_Classes1', 'TPSImport_Classes', iptrw);
21027: ShowIncludel', 'TMenuItem', iptrw);
21028: SynEditPrint1', 'TSynEditPrint', iptrw);
21029: Printout1', 'TMenuItem', iptrw);
21030: mnPrintColors1', 'TMenuItem', iptrw);
21031: dlgFilePrint1', 'TPrintDialog', iptrw);
21032: dlgPrintFont1', 'TFontDialog', iptrw);
21033: mnuPrintFont1', 'TMenuItem', iptrw);
21034: Include1', 'TMenuItem', iptrw);
21035: CodeCompletionList1', 'TMenuItem', iptrw);
21036: IncludeList1', 'TMenuItem', iptrw);
21037: ImageList1', 'TImageList', iptrw);
21038: ImageList2', 'TImageList', iptrw);
21039: CoolBar1', 'TCoolBar', iptrw);
21040: ToolBar1', 'TToolBar', iptrw);
21041: btnLoad', 'TToolButton', iptrw);
21042: ToolButton2', 'TToolButton', iptrw);
21043: btnFind', 'TToolButton', iptrw);
21044: btnCompile', 'TToolButton', iptrw);
21045: btnTrans', 'TToolButton', iptrw);
21046: btnUseCase', 'TToolButton', iptrw); //3.8
21047: toolbtnTutorial', 'TToolButton', iptrw);
21048: btn6res', 'TToolButton', iptrw);
21049: ToolButton5', 'TToolButton', iptrw); 'ToolButton1', 'TToolButton', iptrw);
21050: ToolButton3', 'TToolButton', iptrw); 'statusBar1', 'TStatusBar', iptrw);
21051: SaveOutput1', 'TMenuItem', iptrw); 'ExportClipboard1', 'TMenuItem', iptrw);
21052: Close1', 'TMenuItem', iptrw); 'Manual1', 'TMenuItem', iptrw);
21053: About2', 'TMenuItem', iptrw); 'loadLastfile1', 'TMenuItem', iptrw);
21054: imgLogo', 'TImage', iptrw); 'cedebug', 'TPSScriptDebugger', iptrw);
21055: debugPopupMenu1', 'TPopupMenu', iptrw);
21056: BreakPointMenu', 'TMenuItem', iptrw);
21057: Decompile1', 'TMenuItem', iptrw);
21058: StepInto1', 'TMenuItem', iptrw);
21059: StepOut1', 'TMenuItem', iptrw);
21060: Reset1', 'TMenuItem', iptrw);
21061: DebugRun1', 'TMenuItem', iptrw);
21062: PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
21063: PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
21064: PSImport_Forms1', 'TPSImport_Forms', iptrw);
21065: PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
21066: tutorial4', 'TMenuItem', iptrw);
21067: ExporttoClipboard1', 'TMenuItem', iptrw);
21068: ImportfromClipboard1', 'TMenuItem', iptrw);
21069: N4', 'TMenuItem', iptrw); N5', 'TMenuItem', iptrw); N6', 'TMenuItem', iptrw);
21070: ImportfromClipboard2', 'TMenuItem', iptrw);
21071: tutorial11', 'TMenuItem', iptrw);
21072: N7', 'TMenuItem', iptrw);
21073: ShowSpecChars1', 'TMenuItem', iptrw);
21074: OpenDirectory1', 'TMenuItem', iptrw);
21075: procMess', 'TMenuItem', iptrw);
21076: btnUseCase', 'TToolButton', iptrw);
21077: ToolButton7', 'TToolButton', iptrw);
21078: EditFont1', 'TMenuItem', iptrw);
21079: UseCase1', 'TMenuItem', iptrw);
21080: tutorial21', 'TMenuItem', iptrw);
21081: OpenUseCase1', 'TMenuItem', iptrw);
21082: PSImport_DB1', 'TPSImport_DB', iptrw);
21083: tutorial31', 'TMenuItem', iptrw);
21084: SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
21085: HTMLSyntax1', 'TMenuItem', iptrw);
21086: ShowInterfaces1', 'TMenuItem', iptrw);
21087: Tutorials5', 'TMenuItem', iptrw);
21088: AllFunctionsList1', 'TMenuItem', iptrw);
21089: ShowLastException1', 'TMenuItem', iptrw);
21090: PlayMP31', 'TMenuItem', iptrw);
21091: SynTeXSyn1', 'TSynTeXSyn', iptrw);
21092: texSyntax1', 'TMenuItem', iptrw);
21093: N8', 'TMenuItem', iptrw);
21094: GetEMails1', 'TMenuItem', iptrw);
21095: SynCppSyn1', 'TSynCppSyn', iptrw);
21096: CSyntax1', 'TMenuItem', iptrw);
21097: Tutorial6', 'TMenuItem', iptrw);
21098: New1', 'TMenuItem', iptrw);
21099: AllObjectsList1', 'TMenuItem', iptrw);
21100: LoadBytecode1', 'TMenuItem', iptrw);
21101: CipherFile1', 'TMenuItem', iptrw);
21102: N9', 'TMenuItem', iptrw); N10', 'TMenuItem', iptrw);
21103: Tutorial11', 'TMenuItem', iptrw);
21104: Tutorial71', 'TMenuItem', iptrw);
21105: UpdateService1', 'TMenuItem', iptrw);
21106: PascalSchool1', 'TMenuItem', iptrw);
21107: Tutorial81', 'TMenuItem', iptrw);
21108: DelphiSite1', 'TMenuItem', iptrw);
21109: Output1', 'TMenuItem', iptrw);
21110: TerminalStyle1', 'TMenuItem', iptrw);
21111: ReadOnly1', 'TMenuItem', iptrw);
21112: Shellstyle1', 'TMenuItem', iptrw);
21113: BigScreen1', 'TMenuItem', iptrw);
21114: Tutorial91', 'TMenuItem', iptrw);

```

```
21115: SaveOutput2', 'TMenuItem', iptrw);
21116: N11', 'TMenuItem', iptrw);
21117: SaveScreenshot', 'TMenuItem', iptrw);
21118: Tutorial101', 'TMenuItem', iptrw);
21119: SQLSyntax1', 'TMenuItem', iptrw);
21120: SynSQLSyn1', 'TSynSQLSyn', iptrw);
21121: Console1', 'TMenuItem', iptrw);
21122: SynXMLSyn1', 'TSynXMLSyn', iptrw);
21123: XMLSyntax1', 'TMenuItem', iptrw);
21124: ComponentCount1', 'TMenuItem', iptrw);
21125: NewInstance1', 'TMenuItem', iptrw);
21126: toolbtnTutorial', 'TToolButton', iptrw);
21127: Memory1', 'TMenuItem', iptrw);
21128: SynJavaSyn1', 'TSynJavaSyn', iptrw);
21129: JavaSyntax1', 'TMenuItem', iptrw);
21130: SyntaxCheck1', 'TMenuItem', iptrw);
21131: Tutorial10Statistics1', 'TMenuItem', iptrw);
21132: ScriptExplorer1', 'TMenuItem', iptrw);
21133: FormOutput1', 'TMenuItem', iptrw);
21134: ArduinoDump1', 'TMenuItem', iptrw);
21135: AndroidDump1', 'TMenuItem', iptrw);
21136: GotoEnd1', 'TMenuItem', iptrw);
21137: AllResourceList1', 'TMenuItem', iptrw);
21138: ToolButton4', 'TToolButton', iptrw);
21139: btn6res', 'TToolButton', iptrw);
21140: Tutorial11Forms1', 'TMenuItem', iptrw);
21141: Tutorial12SQL1', 'TMenuItem', iptrw);
21142: ResourceExplore1', 'TMenuItem', iptrw);
21143: Info1', 'TMenuItem', iptrw);
21144: N12', 'TMenuItem', iptrw);
21145: CryptoBox1', 'TMenuItem', iptrw);
21146: Tutorial13Ciphering1', 'TMenuItem', iptrw);
21147: CipherFile2', 'TMenuItem', iptrw);
21148: N13', 'TMenuItem', iptrw);
21149: ModulesCount1', 'TMenuItem', iptrw);
21150: AddOns2', 'TMenuItem', iptrw);
21151: N4GewinntGame1', 'TMenuItem', iptrw);
21152: DocuforAddOns1', 'TMenuItem', iptrw);
21153: Tutorial14Async1', 'TMenuItem', iptrw);
21154: Lessons15Review1', 'TMenuItem', iptrw);
21155: SynPHPSyn1', 'TSynPHPSyn', iptrw);
21156: PHPSyntax1', 'TMenuItem', iptrw);
21157: Breakpoint1', 'TMenuItem', iptrw);
21158: SerialRS2321', 'TMenuItem', iptrw);
21159: N14', 'TMenuItem', iptrw);
21160: SynCSSyn1', 'TSynCSSyn', iptrw);
21161: CSyntax2', 'TMenuItem', iptrw);
21162: Calculator1', 'TMenuItem', iptrw);
21163: btnSerial', 'TToolButton', iptrw);
21164: ToolButton8', 'TToolButton', iptrw);
21165: Tutorial151', 'TMenuItem', iptrw);
21166: N15', 'TMenuItem', iptrw);
21167: N16', 'TMenuItem', iptrw);
21168: ControlBar1', 'TControlBar', iptrw);
21169:ToolBar2', 'TToolBar', iptrw);
21170: BtnOpen', 'TToolButton', iptrw);
21171: BtnSave', 'TToolButton', iptrw);
21172: BtnPrint', 'TToolButton', iptrw);
21173: BtnColors', 'TToolButton', iptrw);
21174: btnClassReport', 'TToolButton', iptrw);
21175: BtnRotateRight', 'TToolButton', iptrw);
21176: BtnFullScreen', 'TToolButton', iptrw);
21177: BtnFitToWindowSize', 'TToolButton', iptrw);
21178: BtnZoomMinus', 'TToolButton', iptrw);
21179: BtnZoomPlus', 'TToolButton', iptrw);
21180: Panel1', 'TPanel', iptrw);
21181: LabelBrettgroesse', ' TLabel', iptrw);
21182: CB1SCList', 'TComboBox', iptrw);
21183: ImageListNormal', 'TImageList', iptrw);
21184: spbtnexployre', 'TSpeedButton', iptrw);
21185: spbtnexample', 'TSpeedButton', iptrw);
21186: spbsaveas', 'TSpeedButton', iptrw);
21187: imglogobox', 'TImage', iptrw);
21188: EnlargeFont1', 'TMenuItem', iptrw);
21189: EnlargeFont2', 'TMenuItem', iptrw);
21190: ShrinkFont1', 'TMenuItem', iptrw);
21191: ThreadDemo1', 'TMenuItem', iptrw);
21192: HEXEditor1', 'TMenuItem', iptrw);
21193: HEXView1', 'TMenuItem', iptrw);
21194: HEXInspect1', 'TMenuItem', iptrw);
21195: SynExporterHTML1', 'TSynExporterHTML', iptrw);
21196: ExporttoHTML1', 'TMenuItem', iptrw);
21197: ClassCount1', 'TMenuItem', iptrw);
21198: HTMLOutput1', 'TMenuItem', iptrw);
21199: HEXEditor2', 'TMenuItem', iptrw);
21200: Minesweeper1', 'TMenuItem', iptrw);
21201: N17', 'TMenuItem', iptrw);
21202: PicturePuzzle1', 'TMenuItem', iptrw);
21203: sbvc1help', 'TSpeedButton', iptrw);
```

```

21204: DependencyWalker1', 'TMenuItem', iptrw);
21205: WebScanner1', 'TMenuItem', iptrw);
21206: View1', 'TMenuItem', iptrw);
21207: mnToolbar1', 'TMenuItem', iptrw);
21208: mnStatusbar2', 'TMenuItem', iptrw);
21209: mnConsole2', 'TMenuItem', iptrw);
21210: mnCoolbar2', 'TMenuItem', iptrw);
21211: mnSplitter2', 'TMenuItem', iptrw);
21212: WebServer1', 'TMenuItem', iptrw);
21213: Tutorial17Server1', 'TMenuItem', iptrw);
21214: Tutorial18Arduino1', 'TMenuItem', iptrw);
21215: SynPerlSyn1', 'TSynPerlSyn', iptrw);
21216: PerlSyntax1', 'TMenuItem', iptrw);
21217: SynPythonSyn1', 'TSynPythonSyn', iptrw);
21218: PythonSyntax1', 'TMenuItem', iptrw);
21219: DMathLibrary1', 'TMenuItem', iptrw);
21220: IntfNavigator1', 'TMenuItem', iptrw);
21221: EnlargeFontConsole1', 'TMenuItem', iptrw);
21222: ShrinkFontConsole1', 'TMenuItem', iptrw);
21223: SetInterfaceList1', 'TMenuItem', iptrw);
21224: popintfList', 'TPopupMenu', iptrw);
21225: intfAdd1', 'TMenuItem', iptrw);
21226: intfDelete1', 'TMenuItem', iptrw);
21227: intfRefactor1', 'TMenuItem', iptrw);
21228: Defactor1', 'TMenuItem', iptrw);
21229: Tutorial19COMArduino1', 'TMenuItem', iptrw);
21230: Tutorial20Regex', 'TMenuItem', iptrw);
21231: N18', 'TMenuItem', iptrw);
21232: ManualE1', 'TMenuItem', iptrw);
21233: FullTextFinder1', 'TMenuItem', iptrw);
21234: Move1', 'TMenuItem', iptrw);
21235: FractalDemo1', 'TMenuItem', iptrw);
21236: Tutorial21Android1', 'TMenuItem', iptrw);
21237: Tutorial0Function1', 'TMenuItem', iptrw);
21238: SimuLogBox1', 'TMenuItem', iptrw);
21239: OpenExamples1', 'TMenuItem', iptrw);
21240: SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
21241: JavaScriptSyntax1', 'TMenuItem', iptrw);
21242: Halt1', 'TMenuItem', iptrw);
21243: CodeSearch1', 'TMenuItem', iptrw);
21244: SynRubySyn1', 'TSynRubySyn', iptrw);
21245: RubySyntax1', 'TMenuItem', iptrw);
21246: Undo1', 'TMenuItem', iptrw);
21247: SynUNIXShellsScriptSyn1', 'TSynUNIXShellsScriptSyn', iptrw);
21248: LinuxShellScript1', 'TMenuItem', iptrw);
21249: Rename1', 'TMenuItem', iptrw);
21250: spdcodesearch', 'TSpeedButton', iptrw);
21251: Preview1', 'TMenuItem', iptrw);
21252: Tutorial22Services1', 'TMenuItem', iptrw);
21253: Tutorial23RealTime1', 'TMenuItem', iptrw);
21254: Configuration1', 'TMenuItem', iptrw);
21255: MP3Player1', 'TMenuItem', iptrw);
21256: DLLSpy1', 'TMenuItem', iptrw);
21257: SynURIOpener1', 'TSynURIOpener', iptrw);
21258: SynURISSyn1', 'TSynURISSyn', iptrw);
21259: URILinksClicks1', 'TMenuItem', iptrw);
21260: EditReplace1', 'TMenuItem', iptrw);
21261: GotoLine1', 'TMenuItem', iptrw);
21262: ActiveLineColor1', 'TMenuItem', iptrw);
21263: ConfigFile1', 'TMenuItem', iptrw);
21264: Sort1Intflist', 'TMenuItem', iptrw);
21265: Redo1', 'TMenuItem', iptrw);
21266: Tutorial24CleanCode1', 'TMenuItem', iptrw);
21267: Tutorial25Configuration1', 'TMenuItem', iptrw);
21268: IndentSelection1', 'TMenuItem', iptrw);
21269: UnindentSection1', 'TMenuItem', iptrw);
21270: SkyStyle1', 'TMenuItem', iptrw);
21271: N19', 'TMenuItem', iptrw);
21272: CountWords1', 'TMenuItem', iptrw);
21273: imbookmarkimages', 'TImageList', iptrw);
21274: Bookmark11', 'TMenuItem', iptrw);
21275: N20', 'TMenuItem', iptrw);
21276: Bookmark21', 'TMenuItem', iptrw);
21277: Bookmark31', 'TMenuItem', iptrw);
21278: Bookmark41', 'TMenuItem', iptrw);
21279: SynMultiSyn1', 'TSynMultiSyn', iptrw);
21280:
21281: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
21282: Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSEExec; x:TPSRuntimeClassImporter);
21283: Procedure PSScriptCompile( Sender : TPSScript)
21284: Procedure Compile1Click( Sender : TObject)
21285: Procedure PSScriptExecute( Sender : TPSScript)
21286: Procedure open1Click( Sender : TObject)
21287: Procedure Save2Click( Sender : TObject)
21288: Procedure Savebefore1Click( Sender : TObject)
21289: Procedure Largefont1Click( Sender : TObject)
21290: Procedure FormActivate( Sender : TObject)
21291: Procedure SBytecode1Click( Sender : TObject)
21292: Procedure FormKeyPress( Sender : TObject; var Key : Char)

```

```

21293: Procedure Saveas3Click( Sender : TObject )
21294: Procedure Clear1Click( Sender : TObject )
21295: Procedure Slinenumbers1Click( Sender : TObject )
21296: Procedure About1Click( Sender : TObject )
21297: Procedure Search1Click( Sender : TObject )
21298: Procedure FormCreate( Sender : TObject )
21299: Procedure Memo1ReplaceText( Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
21300:                                         var Action : TSynReplaceAction)
21301: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges )
21302: Procedure WordWrap1Click( Sender : TObject )
21303: Procedure SearchNext1Click( Sender : TObject )
21304: Procedure Replace1Click( Sender : TObject )
21305: Function PSScriptNeedFile(Sender:TObject;const OriginFileName:String;var FName,Output:String):Bool;
21306: Procedure ShowInclude1Click( Sender : TObject )
21307: Procedure Printout1Click( Sender : TObject )
21308: Procedure mnuPrintFont1Click( Sender : TObject )
21309: Procedure Include1Click( Sender : TObject )
21310: Procedure FormDestroy( Sender : TObject )
21311: Procedure FormClose( Sender : TObject; var Action : TCloseAction )
21312: Procedure UpdateView1Click( Sender : TObject )
21313: Procedure CodeCompletionList1Click( Sender : TObject )
21314: Procedure SaveOutput1Click( Sender : TObject )
21315: Procedure ExportClipboard1Click( Sender : TObject )
21316: Procedure Close1Click( Sender : TObject )
21317: Procedure Manual1Click( Sender : TObject )
21318: Procedure LoadLastFile1Click( Sender : TObject )
21319: Procedure Memo1Change( Sender : TObject )
21320: Procedure Decompile1Click( Sender : TObject )
21321: Procedure StepInto1Click( Sender : TObject )
21322: Procedure StepOut1Click( Sender : TObject )
21323: Procedure Reset1Click( Sender : TObject )
21324: Procedure cedebugAfterExecute( Sender : TPSScript )
21325: Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
21326: Procedure cedebugCompile( Sender : TPSScript )
21327: Procedure cedebugExecute( Sender : TPSScript )
21328: Procedure cedebugIdle( Sender : TObject )
21329: Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal )
21330: Procedure Memo1SpecialLineColor( Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor );
21331: Procedure BreakPointMenuClick( Sender : TObject )
21332: Procedure DebugRun1Click( Sender : TObject )
21333: Procedure tutorial4Click( Sender : TObject )
21334: Procedure ImportfromClipboard1Click( Sender : TObject )
21335: Procedure ImportfromClipboard2Click( Sender : TObject )
21336: Procedure tutorial1Click( Sender : TObject )
21337: Procedure ShowSpecChars1Click( Sender : TObject )
21338: Procedure StatusBar1DblClick( Sender : TObject )
21339: Procedure PSScriptLine( Sender : TObject )
21340: Procedure OpenDirectory1Click( Sender : TObject )
21341: Procedure procMessClick( Sender : TObject )
21342: Procedure tbtnUseCaseClick( Sender : TObject )
21343: Procedure EditFont1Click( Sender : TObject )
21344: Procedure tutorial21Click( Sender : TObject )
21345: Procedure tutorial31Click( Sender : TObject )
21346: Procedure HTMLSyntax1Click( Sender : TObject )
21347: Procedure ShowInterfaces1Click( Sender : TObject )
21348: Procedure Tutorial5Click( Sender : TObject )
21349: Procedure ShowLastException1Click( Sender : TObject )
21350: Procedure PlayMP31Click( Sender : TObject )
21351: Procedure AllFunctionsList1Click( Sender : TObject )
21352: Procedure texSyntax1Click( Sender : TObject )
21353: Procedure GetEMails1Click( Sender : TObject )
21354: procedure DelphiSite1Click(Sender: TObject);
21355: procedure TerminalStyle1Click(Sender: TObject);
21356: procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
21357: procedure ShellStyle1Click(Sender: TObject);
21358: procedure Console1Click(Sender: TObject); //3.2
21359: procedure BigScreen1Click(Sender: TObject);
21360: procedure Tutorial91Click(Sender: TObject);
21361: procedure SaveScreenshotClick(Sender: TObject);
21362: procedure Tutorial101Click(Sender: TObject);
21363: procedure SQLSyntax1Click(Sender: TObject);
21364: procedure XMLSyntax1Click(Sender: TObject);
21365: procedure ComponentCount1Click(Sender: TObject);
21366: procedure NewInstance1Click(Sender: TObject);
21367: procedure CSyntax1Click(Sender: TObject);
21368: procedure Tutorial6Click(Sender: TObject);
21369: procedure New1Click(Sender: TObject);
21370: procedure AllObjectsList1Click(Sender: TObject);
21371: procedure LoadBytecode1Click(Sender: TObject);
21372: procedure CipherFile1Click(Sender: TObject); //V3.5
21373: procedure NewInstance1Click(Sender: TObject);
21374: procedure toolbtnTutorialClick(Sender: TObject);
21375: procedure Memory1Click(Sender: TObject);
21376: procedure JavaSyntax1Click(Sender: TObject);
21377: procedure SyntaxCheck1Click(Sender: TObject);
21378: procedure ScriptExplorer1Click(Sender: TObject);
21379: procedure FormOutput1Click(Sender: TObject); //V3.6
21380: procedure GotoEnd1Click(Sender: TObject);
21381: procedure AllResourceList1Click(Sender: TObject);

```

```

21382: procedure tbtn6resClick(Sender: TObject); //V3.7
21383: procedure Info1Click(Sender: TObject);
21384: procedure Tutorial10Statistics1Click(Sender: TObject);
21385: procedure Tutorial11Forms1Click(Sender: TObject);
21386: procedure Tutorial12SQL1Click(Sender: TObject); //V3.8
21387: procedure ResourceExplore1Click(Sender: TObject);
21388: procedure Info1Click(Sender: TObject);
21389: procedure CryptoBox1Click(Sender: TObject);
21390: procedure ModulesCount1Click(Sender: TObject);
21391: procedure N4GewinntGame1Click(Sender: TObject);
21392: procedure PHPSyntax1Click(Sender: TObject);
21393: procedure SerialRS2321Click(Sender: TObject);
21394: procedure CSyntax2Click(Sender: TObject);
21395: procedure Calculator1Click(Sender: TObject);
21396: procedure Tutorial13Ciphering1Click(Sender: TObject);
21397: procedure Tutorial14Async1Click(Sender: TObject);
21398: procedure PHPSyntax1Click(Sender: TObject);
21399: procedure BtnZoomPlusClick(Sender: TObject);
21400: procedure BtnZoomMinusClick(Sender: TObject);
21401: procedure btnClassReportClick(Sender: TObject);
21402: procedure ThreadDemo1Click(Sender: TObject);
21403: procedure HEXView1Click(Sender: TObject);
21404: procedure ExporttoHTML1Click(Sender: TObject);
21405: procedure Minesweeper1Click(Sender: TObject);
21406: procedure PicturePuzzle1Click(Sender: TObject); //V3.9
21407: procedure sbvc1helpClick(Sender: TObject);
21408: procedure DependencyWalker1Click(Sender: TObject);
21409: procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
21410: procedure WebScanner1Click(Sender: TObject);
21411: procedure mnToolbar1Click(Sender: TObject);
21412: procedure mnStatusbar2Click(Sender: TObject);
21413: procedure mnConsole2Click(Sender: TObject);
21414: procedure mnCoolbar2Click(Sender: TObject);
21415: procedure mnSplitter2Click(Sender: TObject);
21416: procedure WebServer1Click(Sender: TObject);
21417: procedure PerlSyntax1Click(Sender: TObject);
21418: procedure PythonSyntax1Click(Sender: TObject);
21419: procedure DMathLibrary1Click(Sender: TObject);
21420: procedure IntfNavigator1Click(Sender: TObject);
21421: procedure FullTextFinder1Click(Sender: TObject);
21422: function AppName: string;
21423: function ScriptName: string;
21424: function LastName: string;
21425: procedure FractalDemo1Click(Sender: TObject);
21426: procedure SimulogBox1Click(Sender: TObject);
21427: procedure OpenExamples1Click(Sender: TObject);
21428: procedure Halt1Click(Sender: TObject);
21429: procedure Stop;
21430: procedure CodeSearch1Click(Sender: TObject);
21431: procedure RubySyntax1Click(Sender: TObject);
21432: procedure Undo1Click(Sender: TObject);
21433: procedure LinuxShellScript1Click(Sender: TObject);
21434: procedure WebScannerDirect(urls: string);
21435: procedure WebScanner(urls: string);
21436: procedure LoadInterfaceList2;
21437: procedure DLLSpy1Click(Sender: TObject);
21438: procedure Memo1DblClick(Sender: TObject);
21439: procedure URILinksClicks1Click(Sender: TObject);
21440: procedure GotoLine1Click(Sender: TObject);
21441: procedure ConfigFile1Click(Sender: TObject);
21442: Procedure Sort1IntflistClick( Sender : TObject )
21443: Procedure Redo1Click( Sender : TObject )
21444: Procedure Tutorial24CleanCode1Click( Sender : TObject )
21445: Procedure IndentSelection1Click( Sender : TObject )
21446: Procedure UnindentSection1Click( Sender : TObject )
21447: Procedure SkyStyle1Click( Sender : TObject )
21448: Procedure CountWords1Click( Sender : TObject )
21449: Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark )
21450: Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
21451: Procedure Bookmark11Click( Sender : TObject )
21452: Procedure Bookmark21Click( Sender : TObject )
21453: Procedure Bookmark31Click( Sender : TObject )
21454: Procedure Bookmark41Click( Sender : TObject )
21455: Procedure SynMultiSynCustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
21456: 'STATMemoryReport', 'boolean', iptrw;
21457: 'IPPort', 'integer', iptrw;
21458: 'COMPort', 'integer', iptrw;
21459: 'lbintlist', 'TListBox', iptrw;
21460: Function GetStatChange : boolean
21461: Procedure SetStatChange( vstat : boolean )
21462: Function GetActFileName : string
21463: Procedure SetActFileName( vname : string )
21464: Function GetLastFileName : string
21465: Procedure SetLastFileName( vname : string )
21466: Procedure WebScannerDirect( urls : string )
21467: Procedure LoadInterfaceList2
21468: Function GetStatExecuteShell : boolean
21469: Procedure DoEditorExecuteCommand( EditorCommand : word )
21470: function GetActiveLineColor: TColor

```

```

21471: procedure SetActiveLineColor(acolor: TColor)
21472: procedure ScriptListbox1Click(Sender: TObject);
21473: procedure Memo2KeyPress(Sender: TObject; var Key: Char);
21474: procedure EnlargeGutter1Click(Sender: TObject);
21475: procedure Tetris1Click(Sender: TObject);
21476: procedure ToDoList1Click(Sender: TObject);
21477: procedure ProcessList1Click(Sender: TObject);
21478: procedure MetricReport1Click(Sender: TObject);
21479: procedure ProcessList1Click(Sender: TObject);
21480: procedure TCPSockets1Click(Sender: TObject);
21481: procedure ConfigUpdate1Click(Sender: TObject);
21482: procedure ADOWorkbench1Click(Sender: TObject);
21483: procedure SocketServer1Click(Sender: TObject);
21484: procedure FormDemo1Click(Sender: TObject);
21485: procedure Richedit1Click(Sender: TObject);
21486: procedure SimpleBrowser1Click(Sender: TObject);
21487: procedure DOSShell1Click(Sender: TObject);
21488: procedure SynExport1Click(Sender: TObject);
21489: procedure ExporttoRTF1Click(Sender: TObject);
21490: procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
21491: procedure SOAPTester1Click(Sender: TObject);
21492: procedure Sniffer1Click(Sender: TObject);
21493: procedure AutoDetectSyntax1Click(Sender: TObject);
21494: procedure FPlot1Click(Sender: TObject);
21495: procedure PasStyle1Click(Sender: TObject);
21496: procedure Tutorial183RGBLED1Click(Sender: TObject);
21497: procedure ReversilClick(Sender: TObject);
21498: procedure Manualmaxbox1Click(Sender: TObject);
21499: procedure BlaisePascalMagazine1Click(Sender: TObject);
21500: procedure AddToDo1Click(Sender: TObject);
21501: procedure CreateGUID1Click(Sender: TObject);
21502: procedure Tutorial27XML1Click(Sender: TObject);
21503: procedure CreateDLLStub1Click(Sender: TObject);
21504: procedure Tutorial28DLL1Click(Sender: TObject);');
21505: procedure ResetKeyPressed(');
21506: procedure KeyPressedFalse;
21507: procedure FileChanges1Click(Sender: TObject);');
21508: procedure OpenGLTry1Click(Sender: TObject);');
21509: procedure AllUnitList1Click(Sender: TObject);');
21510: procedure Tutorial29UMLClick(Sender: TObject);
21511: procedure CreateHeader1Click(Sender: TObject);
21512: procedure Oscilloscope1Click(Sender: TObject);');
21513: procedure Tutorial30WOT1Click(Sender: TObject);');
21514: procedure GetWebScript1Click(Sender: TObject);');
21515: procedure Checkers1Click(Sender: TObject);');
21516: procedure TaskMgr1Click(Sender: TObject);');
21517: procedure WebCam1Click(Sender: TObject);');
21518: procedure Tutorial31Closure1Click(Sender: TObject);');
21519: procedure GEOMapView1Click(Sender: TObject);');
21520: procedure Run1Click(Sender: TObject);
21521: MaxForm1.GPSSatView1Click, 'GPSSatView1Click');
21522: MaxForm1.N3DLabel1Click, 'N3DLabel1Click');
21523: procedure ExternalApplClick(Sender: TObject);');
21524: procedure PANView1Click(Sender: TObject);
21525: procedure Tutorial39GEOMaps1Click(Sender: TObject);
21526: procedure UnitConverter1Click(Sender: TObject);
21527: maxform1.myscript1click(self)
21528: //-----
21529: //*****mX4 Editor SynEdit Tools API *****
21530: //-----
21531: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
21532: begin //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
21533:   with FindClass('TCustomControl'), 'TCustomSynEdit') do begin
21534:     Constructor Create(AOwner: TComponent)
21535:     SelStart', 'Integer', iptrw);
21536:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
21537:     Procedure UpdateCaret
21538:     Procedure AddKey(Command: TSynEditorCommand; Key1: word; SS1: TShiftState; Key2: word; SS2: TShiftState);
21539:     Procedure AddKey(Command: TSynEditorCommand; Key1: word; SS1: TShiftState; Key2: word; SS2: TShiftState);
21540:     Procedure BeginUndoBlock
21541:     Procedure BeginUpdate
21542:       Function CaretInView: Boolean
21543:       Function CharIndexToRowCol(Index: integer): TBufferCoord
21544:       Procedure Clear
21545:       Procedure ClearAll
21546:       Procedure ClearBookMark(Bookmark: Integer)
21547:       Procedure ClearSelection
21548:       Procedure CommandProcessor(Command: TSynEditorCommand; AChar: char; Data: pointer)
21549:       Procedure ClearUndo
21550:       Procedure CopyToClipboard
21551:       Procedure CutToClipboard
21552:       Procedure DoCopyToClipboard(const SText: string)
21553:       Procedure EndUndoBlock
21554:       Procedure EndUpdate
21555:       Procedure EnsureCursorPosVisible
21556:       Procedure EnsureCursorPosVisibleEx(ForceToMiddle: Boolean)
21557:       Procedure FindMatchingBracket
21558:       Function GetMatchingBracket: TBufferCoord
21559:       Function GetMatchingBracketEx(const APoint: TBufferCoord): TBufferCoord

```

```

21560: Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
21561: Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
21562: Function GetHighlighterAttriAtRowCol( const XY : TBufferCoord; var Token : string; var Attr
21563: : TSynHighlighterAttributes) : boolean
21564: Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
21565: var TokenType, Start : Integer; var Attr:TSynHighlighterAttributes):boolean
21566: Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
21567: Function GetWordAtRowCol( const XY : TBufferCoord) : string
21568: Procedure GotoBookMark( BookMark : Integer)
21569: Procedure GotoLineAndCenter( ALine : Integer)
21570: Function IdentChars : TSynIdentChars
21571: Procedure InvalidateGutter
21572: Procedure InvalidateGutterLine( aLine : integer)
21573: Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
21574: Procedure InvalidateLine( Line : integer)
21575: Procedure InvalidateLines( FirstLine, LastLine : integer)
21576: Procedure InvalidateSelection
21577: Function IsBookmark( BookMark : integer) : boolean
21578: Function IsPointInSelection( const Value : TBufferCoord) : boolean
21579: Procedure LockUndo
21580: Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
21581: Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
21582: Function LineToRow( aLine : integer) : integer
21583: Function RowToLine( aRow : integer) : integer
21584: Function NextWordPos : TBufferCoord
21585: Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
21586: Procedure PasteFromClipboard
21587: Function WordStart : TBufferCoord
21588: Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
21589: Function WordEnd : TBufferCoord
21590: Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
21591: Function PrevWordPos : TBufferCoord
21592: Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
21593: Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
21594: Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
21595: Procedure Redo
21596: Procedure RegisterCommandHandler( const AHandlerProc:THookedCommandEvent; AHandlerData:pointer)
21597: Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
21598: Function RowColToCharIndex( RowCol : TBufferCoord) : integer
21599: Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
21600: Procedure SelectAll
21601: Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
21602: Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
21603: Procedure SetDefaultKeystrokes
21604: Procedure SetSelWord
21605: Procedure SetWordBlock( Value : TBufferCoord)
21606: Procedure Undo Procedure UnlockUndo
21607: Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
21608: Procedure AddKeyUpHandler( aHandler : TKeyEvent)
21609: Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
21610: Procedure AddKeyDownHandler( aHandler : TKeyEvent)
21611: Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
21612: Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
21613: Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
21614: Procedure AddFocusControl( aControl : TWinControl)
21615: Procedure RemoveFocusControl( aControl : TWinControl)
21616: Procedure AddMouseDownHandler( aHandler : TMouseEvent)
21617: Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
21618: Procedure AddMouseUpHandler( aHandler : TMouseEvent)
21619: Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
21620: Procedure AddMouseCursorHandler( aHandler : TMouseEvent)
21621: Procedure RemoveMouseCursorHandler( aHandler : TMouseEvent)
21622: Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
21623: Procedure RemoveLinesPointer
21624: Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
21625: Procedure UnHookTextBuffer
21626: BlockBegin', 'TBufferCoord', iptrw);
21627: BlockEnd', 'TBufferCoord', iptrw);
21628: CanPaste', 'Boolean', iptr); 'CanRedo', 'boolean', iptr);
21629: CanUndo', 'boolean', iptr); 'CaretX', 'Integer', iptrw);
21630: CaretY', 'Integer', iptrw); 'CaretXY', 'TBufferCoord', iptrw);
21631: ActiveLineColor', 'TColor', iptrw);
21632: DisplayX', 'Integer', iptr); 'DisplayY', 'Integer', iptr);
21633: DisplayXY', 'TDisplayCoord', iptr); 'DisplayLineCount', 'integer', iptr);
21634: CharsInWindow', 'Integer', iptr);
21635: CharWidth', 'integer', iptr);
21636: Font', 'TFont', iptrw);
21637: GutterWidth', 'Integer', iptr);
21638: Highlighter', 'TSynCustomHighlighter', iptrw);
21639: LeftChar', 'Integer', iptrw);
21640: LineHeight', 'integer', iptr);
21641: LinesInWindow', 'Integer', iptr);
21642: LineText', 'string', iptrw); Lines', 'TStrings', iptrw);
21643: Marks', 'TSynEditMarkList', iptr);
21644: MaxScrollWidth', 'integer', iptrw);
21645: Modified', 'Boolean', iptrw);
21646: PaintLock', 'Integer', iptr);
21647: ReadOnly', 'Boolean', iptrw);
21648: SearchEngine', 'TSynEditSearchCustom', iptrw);

```

```

21649: SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
21650: SelTabBlock', 'Boolean', iptr);
21651: SelTabLine', 'Boolean', iptr); SelText', 'string', iptrw);
21652: StateFlags', 'TSynStateFlags', iptr);
21653: Text', 'string', iptrw); TopLine', 'Integer', iptrw);
21654: WordAtCursor', 'string', iptr); WordAtMouse', 'string', iptr);
21655: UndoList', 'TSynEditUndoList', iptr);
21656: RedoList', 'TSynEditUndoList', iptr);
21657: OnProcessCommandEvent', 'TProcessCommandEvent', iptrw);
21658: BookMarkOptions', 'TSynBookMarkOpt', iptrw);
21659: BorderStyle', 'TSynBorderStyle', iptrw);
21660: ExtraLineSpacing', 'integer', iptrw);
21661: Gutter', 'TSynGutter', iptrw);
21662: HideSelection', 'boolean', iptrw);
21663: InsertCaret', 'TSynEditCaretType', iptrw);
21664: InsertMode', 'boolean', iptrw); IsScrolling', 'Boolean', iptr);
21665: Keystrokes', 'TSynEditKeyStrokes', iptrw);
21666: MaxUndo', 'Integer', iptrw); Options', 'TSynEditorOptions', iptrw);
21667: OverwriteCaret', 'TSynEditCaretType', iptrw);
21668: RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
21669: ScrollHintColor', 'TColor', iptrw);
21670: ScrollHintFormat', 'TScrollHintFormat', iptrw);
21671: ScrollBars', 'TScrollStyle', iptrw);
21672: SelectedColor', 'TSynSelectedColor', iptrw);
21673: SelectionMode', 'TSynSelectionMode', iptrw);
21674: ActiveSelectionMode', 'TSynSelectionMode', iptrw);
21675: TabWidth', 'integer', iptrw); WantReturns', 'boolean', iptrw);
21676: WantTabs', 'boolean', iptrw); WordWrap', 'boolean', iptrw);
21677: WordWrapGlyph', 'TSynGlyph', iptrw);
21678: OnChange', 'TNotifyEvent', iptrw);
21679: OnClearBookmark', 'TPlaceMarkEvent', iptrw);
21680: OnCommandProcessed', 'TProcessCommandEvent', iptrw);
21681: OnContextHelp', 'TContextHelpEvent', iptrw);
21682: OnDropFiles', 'TDropFilesEvent', iptrw);
21683: OnGutterClick', 'TGutterClickEvent', iptrw);
21684: OnGutterGetText', 'TGutterGetTextEvent', iptrw);
21685: OnGutterPaint', 'TGutterPaintEvent', iptrw);
21686: OnMouseCursor', 'TMouseCursorEvent', iptrw);
21687: OnPaint', 'TPaintEvent', iptrw);
21688: OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
21689: OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
21690: OnReplaceText', 'TReplaceTextEvent', iptrw);
21691: OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
21692: OnStatusChange', 'TStatusChangeEvent', iptrw);
21693: OnPaintTransient', 'TPaintTransient', iptrw);
21694: OnScroll', 'TScrollEvent', iptrw);
21695: end;
21696: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
21697: Function GetPlaceableHighlighters : TSynHighlighterList
21698: Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
21699: Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
21700: Procedure GetEditorCommandValues( Proc : TGetStrProc)
21701: Procedure GetEditorCommandExtended( Proc : TGetStrProc)
21702: Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
21703: Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
21704: Function ConvertCodeStringToExtended( AString : String) : String
21705: Function ConvertExtendedToCodeString( AString : String) : String
21706: Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
21707: Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
21708: Function IndexToEditorCommand( const AIndex : Integer) : Integer
21709:
21710: TSynEditorOption =
21711:   eoAltSetsColumnMode,           //Holding down the Alt Key will put the selection mode into columnar format
21712:   eoAutoIndent,                //Will indent caret on newlines with same amount of leading whitespace as
21713:                           // preceding line
21714:   eoAutoSizeMaxScrollWidth,    //Automatically resizes the MaxScrollWidth property when inserting text
21715:   eoDisableScrollArrows,       //Disables the scroll bar arrow buttons when you can't scroll in that
21716:                           //direction any more
21717:   eoDragDropEditing,          //Allows to select a textblock and drag it in document to another location
21718:   eoDropFiles,                 //Allows the editor accept OLE file drops
21719:   eoEnhanceHomeKey,           //enhances home key positioning, similar to visual studio
21720:   eoEnhanceEndKey,            //enhances End key positioning, similar to JDeveloper
21721:   eoGroupUndo,                 //When undoing/redoing actions, handle all cont.changes same kind in onecall
21722:                           //instead undoing/redoing each command separately
21723:   eoHalfPageScroll,           //By scrolling with page-up/page-down commands,only scroll half page at time
21724:   eoHideShowScrollbars,        //if enabled, then scrollbars will only show if necessary.
21725:   If you have ScrollPasteEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
21726:   eoKeepCaretX,               //When moving through lines w/o cursor Past EOL, keeps X position of cursor
21727:   eoNoCaret,                  //Makes it so the caret is never visible
21728:   eoNoSelection,              //Disables selecting text
21729:   eoRightMouseMovesCursor,    //When clicking with right mouse for popup menu, moves cursor to location
21730:   eoScrollByOneLess,          //Forces scrolling to be one less
21731:   eoScrollHintFollows,         //The scroll hint follows the mouse when scrolling vertically
21732:   eoScrollPastEof,            //Allows the cursor to go past the end of file marker
21733:   eoScrollPastEol,             //Allows cursor to go past last character into white space at end of a line
21734:   eoShowScrollHint,            //Shows a hint of the visible line numbers when scrolling vertically
21735:   eoShowSpecialChars,          //Shows the special Characters
21736:   eoSmartTabDelete,            //similar to Smart Tabs, but when you delete characters
21737:   eoSmartTabs,                //When tabbing, cursor will go to non-white space character of previous line

```

```

21738: eoSpecialLineDefaultFg,      //disables the foreground text color override using OnSpecialLineColor event
21739: eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
21740: eoTabsToSpaces,             //Converts a tab character to a specified number of space characters
21741: eoTrimTrailingSpaces     //Spaces at the end of lines will be trimmed and not saved
21742:
21743: *****Important Editor Short Cuts*****
21744: Double click to select a word and count words with highlightning.
21745: Triple click to select a line.
21746: CTRL+SHIFT+click to extend a selection.
21747: Drag with the ALT key down to select columns of text !!!
21748: Drag and drop is supported.
21749: Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
21750: Type CTRL+A to select all.
21751: Type CTRL+N to set a new line.
21752: Type CTRL+T to delete a line or token. //Tokenizer
21753: Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
21754: Type CTRL+Shift+T to add ToDo in line and list.
21755: Type CTRL+Shift+[0..9] to set bookmarks. //Bookmark
21756: Type CTRL[0..9] to jump or get to bookmarks.
21757: Type Home to position cursor at beginning of current line and End to position it at end of line.
21758: Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
21759: Page Up and Page Down work as expected.
21760: CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
21761: using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
21762:
21763: { $ Short Key Positions Ctrl<A-Z>: }
21764: def
21765:   <A> Select All
21766:   <B> Count Words
21767:   <C> Copy
21768:   <D> Internet Start
21769:   <E> Script List
21770:   <F> Find
21771:   <G> Goto
21772:   <H> Mark Line
21773:   <I> Interface List
21774:   <J> Code Completion
21775:   <K> Console
21776:   <L> Interface List Box
21777:   <M> Font Smaller -
21778:   <N> New Line
21779:   <O> Open File
21780:   <P> Font Larger +
21781:   <Q> Quit
21782:   <R> Replace
21783:   <S> Save!
21784:   <T> Delete Line
21785:   <U> Use Case Editor
21786:   <V> Paste
21787:   <W> URI Links
21788:   <X> Reserved for coding use internal
21789:   <Y> Delete Line
21790:   <Z> Undo
21791:
21792: ref F1 Help
21793:   F2 Syntax Check
21794:   F3 Search Next
21795:   F4 New Instance
21796:   F5 Line Mark /Breakpoint
21797:   F6 Goto End
21798:   F7 Debug Step Into
21799:   F8 Debug Step Out
21800:   F9 Compile
21801:   F10 Menu
21802:   F11 Word Count Highlight
21803:   F12 Reserved for coding use internal
21804:
21805: AddRegisteredVariable('Application', 'TApplication');
21806: AddRegisteredVariable('Screen', 'TScreen');
21807: AddRegisteredVariable('Self', 'TForm');
21808: AddRegisteredVariable('Mem1', 'TSynMemo');
21809: AddRegisteredVariable('memo2', 'TMemo');
21810: AddRegisteredVariable('maxForm1', 'TMaxform1'); //!
21811: AddRegisteredVariable('debugout', 'Tdebugoutput'); //!
21812: AddRegisteredVariable('hlog', 'THotlog'); //!
21813: AddRegisteredVariable( it ,integer'); //for closure!!
21814: AddRegisteredVariable( sr ,string'); //for closure
21815: AddRegisteredVariable( bt ,boolean'); //for closure
21816: AddRegisteredVariable( ft ,double'); //for closure
21817: AddRegisteredVariable( srlist , TStringlist'); //for closures
21818:
21819: def ReservedWords: array[0..86] of string =
21820:   ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
21821:    'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
21822:    'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
21823:    'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
21824:    'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
21825:    'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
21826:    'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',

```

```

21827:     'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
21828:     'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
21829:     'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
21830:     'public', 'published', def.ref.using,typedef ,memo1', 'memo2', 'doc', 'maxform1', 'it';
21831:     AllowedChars: array[0..5] of string = ('(', ')', '[', ']', ',', ', t,t1,t2,t3: boolean;
21832: //-----
21833: //*****End of mX4 Public Tools API *****
21834: //-----
21835:
21836: maxbox Internal Inventory
21837: Amount of Functions: 15031
21838: Amount of Procedures: 9172
21839: Amount of Constructors: 1492
21840: Totals of Calls: 25695
21841: SHA1: Win of 3.9.9.195 F0AB7D054111F5CE46BA122D6280397A841C6FAB
21842:
21843: ****
21844: Doc Short Manual with 50 Tips!
21845: ****
21846: - Install: just save your maxboxdef.ini before and then extract the zip file!
21847: - Toolbar: Click on the red maxbox Sign (right on top) opens your work directory or jump to <Help>
21848: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
21849: - Menu: With <Ctrl><F3> you can search for code on examples
21850: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
21851: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
21852: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
21853:
21854: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
21855: - Inifile: Refresh (reload) the inifile after edit with ..//Help/Config Update
21856: - Context Menu: You can printout your scripts as a pdf-file or html-export
21857: - Context: You do have a context menu with the right mouse click
21858:
21859: - Menu: With the UseCase Editor you can convert graphic formats too.
21860: - Menu: On menu Options you find Addons as compiled scripts
21861: - IDE: Menu Program: Run Only is faster, after F2 - You don't need a mouse use shortcuts
21862: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
21863: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
21864: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
21865:         or ttimer (you type classp and then CTRL J), or you type tstringlist and <Ctrl><J>
21866:
21867: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
21868: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
21869: - Code: If you code a loop till key-pressed use function: isKeyPressed;
21870: - Code: Macro set the macros #name,, #paAdministrator, #file,startmaxbox_extract_funcList399.txt
21871: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
21872: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
21873:         to delete and Click and mark to drag a bookmark
21874: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
21875: - IDE: A file info with system and script information you find in menu Program/Information
21876: - IDE: After change the config file in help you can update changes in menu Help/Config Update
21877: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
21878: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
21879: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
21880: - Editor: Set Bookmarks to check your work in app or code
21881: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
21882: - Editor: With //TODO: some description or DONE you set code entries for ToDo List in ..//Help/ToDo List
21883: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
21884: - IDE with menu /Options/ADO SQL Workbench you can manage your Database
21885: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
21886: - Menu: Set Interface Naviagator also with toggle <Ctrl L> or /View/Intf Navigator
21887: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
21888: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
21889: - Code Editor: Compile with <F9> but also Alt C in case <F9> isn't available;
21890: - Code: if you cant run a function try the second one, for ex. Voice() - Voice2(), inc() - incl()
21891: - IDE set bookmarks with <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
21892: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
21893: - IDE menu /Help/Tools/ open the Task Manager
21894:
21895: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
21896: - Add on when no browser is available start /Options/Add ons/Easy Brower
21897: - Add on SOAP Tester with SOP POST File
21898: - Add on IP Protocol Sniffer with List View
21899: - Add on OpenGL mX Robot Demo for android
21900: - Add on Checkers Game, Add on Oscilloscope /View GEO Map View3
21901:
21902: - Menu: Help/Tools as a Tool Section with DOS Opener
21903: - Menu Editor: export the code as RTF File
21904: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
21905: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
21906: - Context: Auto Detect of Syntax depending on file extension
21907: - Code: some Windows API function start with w in the name like wGetAtomName();
21908: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
21909: - IDE File Check with menu ..View/File Changes/...
21910: - Context: Create a Header with Create Header in Navigator List at right window
21911: - Code: use SysErrorMessage to get a real Error Description, Ex.
21912:     RemoveDir('c:\NoSuchFolder'); writeln('System Error Message:' + SysErrorMessage(GetLastError));
21913: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
21914: - Editor: with <Ctrl W> you can click on hyperlinks in Code - CTRL Click on link
21915: - Editor: with <Ctrl+Alt+R> you can write in RTF format with RichEdit link

```

```

21916: - Menu: Check Help/Tools! you can use richedit, DOS Shell or Explorer
21917: - Menu: Start View/MyScript of ini file maxboxdef.ini [MYSCRIPT]= path of script
21918:
21919: - using DLL example in maXbox: //function: {*****}
21920:   Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
21921:                                     cb: DWORD): BOOL; //stdcall;;
21922:   External 'GetProcessMemoryInfo@psapi.dll stdcall';
21923:   Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
21924:     External 'OpenProcess@kernel32.dll stdcall';
21925:
21926: GCC Compile Ex Script
21927: procedure TFormMain_btnCompileClick(Sender: TObject);
21928: begin
21929:   AProcess:= TProcess.Create(Nil);
21930:   try AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
21931:     + ' -o "' + OpenDialog2.FileName + '"';
21932:   AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
21933:   AProcess.Execute;
21934:   Memo2.Lines.BeginUpdate;
21935:   Memo2.Lines.Clear;
21936:   Memo2.Lines.LoadFromStream(AProcess.Output);
21937:   Memo2.Lines.EndUpdate;
21938: finally
21939:   AProcess.Free;
21940: end;
21941: end;
21942:
21943: ref Stopwatch pattern snip
21944: Time1:= Time;
21945: writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
21946: if initAndStartBoard then
21947:   writeln('Filesize: '+inttostr(filesize(FILESAVE)));
21948: writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
21949: PrintF('%d %s',[Trunc((Time-Time1)*24),FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
21950:
21951: POST git-receive-pack (chunked)
21952: Pushing to https://github.com/maxkleiner/maXbox3.git
21953: To https://github.com/maxkleiner/maXbox3.git f127d21..c6a98da masterbox2 -> masterbox2
21954: updating local tracking ref 'refs/remotes/maxkleiner/maXbox3Remote/masterbox2'
21955:
21956: History Shell Hell - Walk the Talk
21957: PCT Precompile Technology , mX4 ScriptStudio
21958: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
21959: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
21960: emax layers: system-package-component-unit-class-function-block
21961: new keywords def ref using maXCalcF UML: use case act class state seq pac comp dep - lib lab
21962: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
21963: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
21964: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
21965: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
21966: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
21967: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
21968: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
21969: SendBuffer, Color+Caption hack, ComboSet, SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
21970: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
21971: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
21972: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21973: Inno Install and Setup Routines Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
21974: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21975: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
21976: 9 Color LED, LED Resources, Runtime LED, it + sr var, morse generator
21977: Add 5 Units, 1 Tutors, maxmap, OpenStreetView, MAPX
21978: Function Menu/View/GEO Map View, DownloadFile, wgetX, sensors
21979: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
21980: ByteCode2, IPUtills2, GEOCode, CGI-Powtils, GPS_2, External App, Unit Converter
21981: Add 16 Units, 1 Slide, Tutor, Big Numbers (Decimals, TInteger), ModBusTCP, TGEOInfo
21982: Add 36 Units, 1 Tutor, SOAPConn, AVI Res, OLEUtils, ACM, CDS, XMLDoc, DDE
21983:
21984: Ref:
21985: https://unibe-ch.academia.edu/MaxKleiner
21986: http://www.slideshare.net/maxkleiner1
21987: http://www.scribd.com/max_kleiner
21988: http://www.delphiforfun.org/Programs/Utilities/index.htm
21989: http://www.slideshare.net/maxkleiner1
21990: http://s3.amazonaws.com/PreviewLinks/22959.html
21991: http://www.softwareschule.ch/arduino_training.pdf
21992: http://www.jrsoftware.org/isinfo.php
21993: http://www.be-precision.com/products/precision-builder/express/
21994: http://www.blaisepascal.eu/
21995: http://www.delhibasics.co.uk/
21996: http://www.youtube.com/watch?v=av89HAbqAsI
21997: http://www.angelfire.com/his5/delphizeus/modal.html
21998: http://www.retrowarehouse.org/garbo/po/turbopas/index.html
21999: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
22000: http://delphi.org/2014/01/every-android-api-for-delphi/
22001: https://en.wikipedia.org/wiki/User:Maxkleiner
22002: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
22003: https://bitbucket.org/max_kleiner/maxbox3
22004: https://bitbucket.org/max_kleiner/maxbox3/downloads

```

```

22005: https://bitbucket.org/max_kleiner/maxbox3/wiki/maxbox%20Tutorials
22006: http://www.slideshare.net/maxkleiner1/codereview-topics
22007:
22008: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
22009: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
22010: =renderBasicSearchNarrative&q=%s';
22011: UrlMapQuestAPIReverse='http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
22012: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
22013:
22014: function OpenMap(const Data: string): boolean;
22015: var encURL: string;
22016: begin
22017:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPEncode(Data)]);
22018:   try //HttpGet(EncodedURL, mapStream); //WinInet
22019:     Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
22020:   //OpenDoc(Exepath+'openmapx.html');
22021:   S_ShellExecute(Exepath+'openmapx.html','','seCmdOpen');
22022:   finally
22023:     encURL:= '';
22024:   end;
22025: end;
22026:
22027: procedure GetGEOMap(C_form,apath: string; const Data: string);
22028: var encodedURL: string; mapStream: TMemoryStream;
22029: begin //encodedURL:= Format(UrlGoogleQrCode,[Width,Height, C_Level, HTTPEncode(Data)]);
22030:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
22031:   mapStream:= TMemoryStream.create;
22032:   try
22033:     Wininet_HttpGet(EncodedURL, mapStream); //WinInet
22034:     mapStream.Position:= 0;
22035:     mapStream.Savetofile(apath); // OpenDoc(apath);
22036:     S_ShellExecute(apath,'',seCmdOpen);
22037:   finally
22038:     mapStream.Free;
22039:   end;
22040: end;
22041:
22042: Procedure BtnFactory(a,b,c,d:smallint; title,apic:string;
22043:                         var abtn:TBitBtn; anEvent:TNotifyEvent; afrm:TForm);
22044: begin
22045:   abtn:= TBitBtn.create(afrm);
22046:   with abtn do begin
22047:     parent:= afrm;
22048:     setBounds(a,b,c,d);
22049:     font.size:= 12;
22050:     glyph.LoadFromResourceName(HINSTANCE, apic);
22051:     mxButton(5,5,width, height,12,12,handle);
22052:     caption:= title;
22053:     onClick:= anEvent As TNotifyEvent;
22054:   end;
22055: end;
22056:
22057: function MySoundcard: Longint; external 'waveOutGetNumDevs@winmm.dll stdcall';
22058: function isSound: boolean; begin result:= mySoundcard > 0 end;
22059: function StringtoHex(Data: string): string;
22060: function GetAnsistringRefCount(const S: string): Cardinal;
22061: function WideStringToString(const ws: WideString; codePage: Word): AnsiString;
22062: function StringToWideString(const s: AnsiString; codePage: Word): WideString;
22063: procedure FreeObjectList(List: TObjectList);
22064: function SecondToTime(const Seconds: Cardinal): Double;
22065: function CopyDir2(const fromDir, toDir: string): Boolean;';
22066: function MoveDir(const fromDir, toDir: string): Boolean;';
22067: function DelDir(dir: string): Boolean;';
22068: procedure DeleteScansRect(Src, Dest: TBitmap; rs, rd: TRect);
22069: procedure FadeIn(ImageFileName: TFileName; aForm1: TForm);
22070: procedure FadeOut(ImageFileName: TFileName);
22071: procedure FadeOut32(const Bmp: TImage; Pause: Integer);
22072: function CheckBDEInstalled: Boolean; //IsBDE
22073: Function GetNumberOfEventLogRecords(hEventLog: THandle; var NumberOfRecords: DWORD): BOOL;
22074: Function GetOldestEventLogRecord( hEventLog : THHandle; var OldestRecord : DWORD ) : BOOL';
22075: ex.: EventLog:= RegisterEventSource('0','maxbox3.exe'));
22076:   if GetNumberOfEventLogRecords(eventlog, recs) then
22077:
22078: function ListIdentical2(l1,l2:TStringList): Boolean;
22079: begin Result:= False;
22080:   if l1.count = l2.count then begin
22081:     for it:= 0 to l1.count-1 do
22082:       if (l1[it] <> l2[it]) then Exit;
22083:     Result:= True;
22084:   end;
22085: end;
22086:
22087: // Converts String To Hexadecimal
22088: // Maybe usefull for a hex-editor
22089: // For example: Input = 'ABCD' Output = '41 42 43 44'
22090:
22091: function StringtoHex(Data: string): string;
22092: var
22093:   i, i2: Integer; s: string;

```

```

22094: begin
22095:   i2 := 1;
22096:   for i := 1 to Length(Data) do begin
22097:     Inc(i2);
22098:     if i2 = 2 then begin
22099:       s := s + ' ';
22100:       i2 := 1;
22101:     end;
22102:     s := s + IntToHex(Ord(Data[i]), 2);
22103:   end;
22104:   Result := s;
22105: end;
22106:
22107:
22108: ****
22109: unit List asm internal end
22110: ****
22111: 01 unit RIRegister_Utils_Routines(exec); //Delphi
22112: 02 unit SIRegister_IdStrings; //Indy Sockets
22113: 03 unit RIRegister_niSTRING_Routines(Exec); //from RegEx
22114: 04 unit uPSI_fMain_Functions; //maXbox Open Tools API
22115: 05 unit IFSI_WinFormlpuzzle; //maXbox
22116: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImagefileLibBCB
22117: 07 unit RegisterDateTimeLibrary_R(exec); //Delphi
22118: 08 unit RIRegister_MathMax_Routines(exec); //Jedi & Delphi
22119: 09 unit RIRegister_IdGlobal_Routines(exec); //Indy Sockets
22120: 10 unit RIRegister_SysUtils_Routines(Exec); //Delphi
22121: 11 unit uPSI_IdTCPConnection; //Indy some functions
22122: 12 unit uPSCompiler.pas; //PS kernel functions
22123: 13 unit uPSI_DBCommon; //DB Common_Routines and Types
22124: 14 unit uPSI_Printers.pas; //Delphi VCL
22125: 15 unit uPSI_MPlayer.pas; //Delphi VCL
22126: 16 unit uPSC_comobj; //COM Functions
22127: 17 unit uPSI_Clipbrd; //Delphi VCL
22128: 18 unit Filectrl in IFSI_SysUtils_max; //VCL Runtime
22129: 19 unit uPSI_SQLExpr; //DBX3
22130: 20 unit uPSI_ADOdb; //ADODB
22131: 21 unit uPSI_StrHlpr; //String Helper Routines
22132: 22 unit uPSI_DateUtils; //Expansion to DateTimeLib
22133: 23 unit uPSI_Fileutils; //Expansion to Sys/File Utils
22134: 24 unit JUutils / gsUtils; //Jedi / Metabase
22135: 25 unit JvFunctions_max; //Jedi Functions
22136: 26 unit HTTPParser; //Delphi VCL
22137: 27 unit HTTPUtil; //Delphi VCL
22138: 28 unit uPSI_XMLUtil; //Delphi VCL
22139: 29 unit uPSI_SOAPHTTPClient; //Delphi VCL SOAP WebService V3.5
22140: 30 unit uPSI_Contrns; //Delphi RTL Container of Classes
22141: 31 unit uPSI_MaskUtils; //RTL Edit and Mask functions
22142: 32 unit uPSI_MyBigInt; //big integer class with Math
22143: 33 unit uPSI_ConvUtils; //Delphi VCL Conversions engine
22144: 34 unit Types_Variants; //Delphi\Win32\rtl\sys
22145: 35 unit uPSI_IdHashSHA1; //Indy Crypto Lib
22146: 36 unit uPSI_IdHashMessageDigest; //Indy Crypto;
22147: 37 unit uPSI_IdASN1Util; //Indy ASN1Utility Routines;
22148: 38 unit uPSI_IdLogFile; //Indy Logger from LogBase
22149: 39 unit uPSI_IdICmpClient; //Indy Ping ICMP
22150: 40 unit uPSI_IdHashMessageDigest_max; //Indy Crypto &OpenSSL;
22151: 41 unit uPSI_FileCtrl; //Delphi RTL
22152: 42 unit uPSI_Outline; //Delphi VCL
22153: 43 unit uPSI_ScktComp; //Delphi RTL
22154: 44 unit uPSI_Calendar; //Delphi VCL
22155: 45 unit uPSI_VListView; //VListView;
22156: 46 unit uPSI_DBGrids; //Delphi VCL
22157: 47 unit uPSI_DBCtrls; //Delphi VCL
22158: 48 unit ide_debugoutput; //maXbox
22159: 49 unit uPSI_ComCtrls; //Delphi VCL
22160: 50 unit uPSC_stdCtrls+; //Delphi VCL
22161: 51 unit uPSI_Dialogs; //Delphi VCL
22162: 52 unit uPSI_StdConvs; //Delphi RTL
22163: 53 unit uPSI_DBClient; //Delphi RTL
22164: 54 unit uPSI_DBPlatform; //Delphi RTL
22165: 55 unit uPSI_Provider; //Delphi RTL
22166: 56 unit uPSI_FMTBcd; //Delphi RTL
22167: 57 unit uPSI_DBCGrids; //Delphi VCL
22168: 58 unit uPSI_CDSSUtil; //MIDAS
22169: 59 unit uPSI_VarHlpr; //Delphi RTL
22170: 60 unit uPSI_ExtDlgs; //Delphi VCL
22171: 61 unit sdpStopwatch; //maXbox
22172: 62 unit uPSI_JclStatistics; //JCL
22173: 63 unit uPSI_JclLogic; //JCL
22174: 64 unit uPSI_JclMiscel; //JCL
22175: 65 unit uPSI_JclMath_max; //JCL RTL
22176: 66 unit uPSI_uTPLb_StreamUtils; //LockBox 3
22177: 67 unit uPSI_MathUtils; //BCB
22178: 68 unit uPSI_JclMultimedia; //JCL
22179: 69 unit uPSI_WideStrUtils; //Delphi API/RTL
22180: 70 unit uPSI_GraphUtil; //Delphi RTL
22181: 71 unit uPSI_TypeTrans; //Delphi RTL
22182: 72 unit uPSI_HTTPApp; //Delphi VCL

```

```

22183: 73 unit uPSI_DBWeb;                                //Delphi VCL
22184: 74 unit uPSI_DBBdeWeb;                            //Delphi VCL
22185: 75 unit uPSI_DBXpressWeb;                          //Delphi VCL
22186: 76 unit uPSI_ShadowWnd;                            //Delphi VCL
22187: 77 unit uPSI_ToolWin;                             //Delphi VCL
22188: 78 unit uPSI_Tabs;                               //Delphi VCL
22189: 79 unit uPSI_JclGraphUtils;                      //JCL
22190: 80 unit uPSI_JclCounter;                           //JCL
22191: 81 unit uPSI_JclSysInfo;                           //JCL
22192: 82 unit uPSI_JclSecurity;                          //JCL
22193: 83 unit uPSI_JclFileUtils;                         //JCL
22194: 84 unit uPSI_IdUserAccounts;                      //Indy
22195: 85 unit uPSI_IdAuthentication;                   //Indy
22196: 86 unit uPSI_uTPLb_AES;                            //LockBox 3
22197: 87 unit uPSI_IdHashSHA1;                           //LockBox 3
22198: 88 unit uTPLb_BlockCipher;                         //LockBox 3
22199: 89 unit uPSI_ValEdit.pas;                          //Delphi VCL
22200: 90 unit uPSI_JvVCLUtils;                           //JCL
22201: 91 unit uPSI_JvDBUtil;                            //JCL
22202: 92 unit uPSI_JvDBUtils;                            //JCL
22203: 93 unit uPSI_JvAppUtils;                           //JCL
22204: 94 unit uPSI_JvCtrlUtils;                          //JCL
22205: 95 unit uPSI_JvFormToHtml;                         //JCL
22206: 96 unit uPSI_JvParsing;                            //JCL
22207: 97 unit uPSI_SerDlg;                             //Toolbox
22208: 98 unit uPSI_Serial;                             //Toolbox
22209: 99 unit uPSI_JvComponent;                          //JCL
22210: 100 unit uPSI_JvCalc;                            //JCL
22211: 101 unit uPSI_JvBdeUtils;                          //JCL
22212: 102 unit uPSI_JvDateUtil;                          //JCL
22213: 103 unit uPSI_JvGenetic;                           //JCL
22214: 104 unit uPSI_JclBase;                            //JCL
22215: 105 unit uPSI_JvUtils;                            //JCL
22216: 106 unit uPSI_JvStringUtil;                        //JCL
22217: 107 unit uPSI_JvStringUtil;                        //JCL
22218: 108 unit uPSI_JvFileUtil;                           //JCL
22219: 109 unit uPSI_JvMemoryInfos;                      //JCL
22220: 110 unit uPSI_JvComputerInfo;                     //JCL
22221: 111 unit uPSI_JvgCommClasses;                    //JCL
22222: 112 unit uPSI_JvgLogics;                           //JCL
22223: 113 unit uPSI_JvLED;                             //JCL
22224: 114 unit uPSI_JvTurtle;                            //JCL
22225: 115 unit uPSI_SortThds; unit uPSI_ThSort;        //maxbox
22226: 116 unit uPSI_JvgUtils;                           //JCL
22227: 117 unit uPSI_JvExprParser;                        //JCL
22228: 118 unit uPSI_HexDump;                            //Borland
22229: 119 unit uPSI_DBLogDlg;                           //VCL
22230: 120 unit uPSI_SqlTimSt;                           //RTL
22231: 121 unit uPSI_JvHtmlParser;                        //JCL
22232: 122 unit uPSI_JvgXMLSerializer;                  //JCL
22233: 123 unit uPSI_JvJCLUtils;                          //JCL
22234: 124 unit uPSI_JvStrings;                           //JCL
22235: 125 unit uPSI_uTPLb_IntegerUtils;                 //TurboPower
22236: 126 unit uPSI_uTPLb_HugeCardinal;                //TurboPower
22237: 127 unit uPSI_uTPLb_HugeCardinalUtils;           //TurboPower
22238: 128 unit uPSI_SynRegExpr;                          //SynEdit
22239: 129 unit uPSI_StUtils;                            //SysTools4
22240: 130 unit uPSI_StToHTML;                           //SysTools4
22241: 131 unit uPSI_StStrms;                           //SysTools4
22242: 132 unit uPSI_StFIN;                            //SysTools4
22243: 133 unit uPSI_StAstroP;                           //SysTools4
22244: 134 unit uPSI_StStat;                            //SysTools4
22245: 135 unit uPSI_StNetCon;                           //SysTools4
22246: 136 unit uPSI_StDecMth;                           //SysTools4
22247: 137 unit uPSI_StOStr;                            //SysTools4
22248: 138 unit uPSI_StPtnrs;                           //SysTools4
22249: 139 unit uPSI_StNetMsg;                           //SysTools4
22250: 140 unit uPSI_StMath;                            //SysTools4
22251: 141 unit uPSI_StExpEng;                           //SysTools4
22252: 142 unit uPSI_StCRC;                            //SysTools4
22253: 143 unit uPSI_StExport;                           //SysTools4
22254: 144 unit uPSI_StExpLog;                           //SysTools4
22255: 145 unit uPSI_ActnList;                           //Delphi VCL
22256: 146 unit uPSI_jpeg;                             //Borland
22257: 147 unit uPSI_StRandom;                           //SysTools4
22258: 148 unit uPSI_StDict;                            //SysTools4
22259: 149 unit uPSI_StBCD;                            //SysTools4
22260: 150 unit uPSI_StTxtDat;                           //SysTools4
22261: 151 unit uPSI_StRegEx;                           //SysTools4
22262: 152 unit uPSI_IMouse;                            //VCL
22263: 153 unit uPSI_SyncObjs;                          //VCL
22264: 154 unit uPSI_AsyncCalls;                        //Hausladen
22265: 155 unit uPSI_ParallelJobs;                      //Saraiva
22266: 156 unit uPSI_Variants;                           //VCL
22267: 157 unit uPSI_VarCmplx;                           //VCL Wolfram
22268: 158 unit uPSI_DTDSchema;                          //VCL
22269: 159 unit uPSI_ShLwApi;                            //Brakel
22270: 160 unit uPSI_IBUtils;                            //VCL
22271: 161 unit uPSI_CheckLst;                           //VCL

```

```

22272: 162 unit uPSI_JvSimpleXml; //JCL
22273: 163 unit uPSI_JclSimpleXml; //JCL
22274: 164 unit uPSI_JvXmlDatabase; //JCL
22275: 165 unit uPSI_JvMaxPixel; //JCL
22276: 166 unit uPSI_JvItemsSearchs; //JCL
22277: 167 unit uPSI_StExpEng2; //SysTools4
22278: 168 unit uPSI_StGenLog; //SysTools4
22279: 169 unit uPSI_JvLogFile; //Jcl
22280: 170 unit uPSI_CPort; //ComPort Lib v4.11
22281: 171 unit uPSI_CPortCtl; //ComPort
22282: 172 unit uPSI_CPortEsc; //ComPort
22283: 173 unit BarCodeScanner; //ComPort
22284: 174 unit uPSI_JvGraph; //JCL
22285: 175 unit uPSI_JvComCtrls; //JCL
22286: 176 unit uPSI_GUITesting; //D Unit
22287: 177 unit uPSI_JvFindFiles; //JCL
22288: 178 unit uPSI_StSystem; //SysTools4
22289: 179 unit uPSI_JvKeyboardStates; //JCL
22290: 180 unit uPSI_JvMail; //JCL
22291: 181 unit uPSI_JclConsole; //JCL
22292: 182 unit uPSI_JclLANMan; //JCL
22293: 183 unit uPSI_IdCustomHTTPServer; //Indy
22294: 184 unit IdHTTPServer //Indy
22295: 185 unit uPSI_IdTCPServer; //Indy
22296: 186 unit uPSI_IdSocketHandle; //Indy
22297: 187 unit uPSI_IdIOHandlerSocket; //Indy
22298: 188 unit IdIOHandler; //Indy
22299: 189 unit uPSI_ututils; //Bloodshed
22300: 190 unit uPSI-BoldUtils; //boldsoft
22301: 191 unit uPSI_IdSimpleServer; //Indy
22302: 192 unit uPSI_IdSSLOpenSSL; //Indy
22303: 193 unit uPSI_IdMultipartFormData; //Indy
22304: 194 unit uPSI_SynURIOpener; //SynEdit
22305: 195 unit uPSI_PerlRegEx; //PCRE
22306: 196 unit uPSI_IdHeaderList; //Indy
22307: 197 unit uPSI_StFirst; //SysTools4
22308: 198 unit uPSI_JvCtrls; //JCL
22309: 199 unit uPSI_IdTrivialFTPBase; //Indy
22310: 200 unit uPSI_IdTrivialFTP; //Indy
22311: 201 unit uPSI_IdUDPBase; //Indy
22312: 202 unit uPSI_IdUDPClient; //Indy
22313: 203 unit uPSI_utypes; //for DMath.DLL
22314: 204 unit uPSI_ShellAPI; //Borland
22315: 205 unit uPSI_IdRemoteCMDClient; //Indy
22316: 206 unit uPSI_IdRemoteCMDServer; //Indy
22317: 207 unit IdRexecServer; //Indy
22318: 208 unit IdRexec; (unit uPSI_IdRexec;) //Indy
22319: 209 unit IdUDPServer; //Indy
22320: 210 unit IdTimeUDPServer; //Indy
22321: 211 unit IdTimeServer; //Indy
22322: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
22323: 213 unit uPSI_IdIPWatch; //Indy
22324: 214 unit uPSI_IdIrcServer; //Indy
22325: 215 unit uPSI_IdMessageCollection; //Indy
22326: 216 unit uPSI_cPEM; //Fundamentals 4
22327: 217 unit uPSI_cFundamentUtils; //Fundamentals 4
22328: 218 unit uPSI_uwinplot; //DMath
22329: 219 unit uPSI_xrtl_util_CPUUtils; //ExtendedRTL
22330: 220 unit uPSI_GR32_System; //Graphics32
22331: 221 unit uPSI_cFileUtils; //Fundamentals 4
22332: 222 unit uPSI_cDateTime; (timemachine) //Fundamentals 4
22333: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
22334: 224 unit uPSI_cRandom; //Fundamentals 4
22335: 225 unit uPSI_ueval; //DMath
22336: 226 unit uPSI_xrtl_net_URIUtils; //ExtendedRTL
22337: 227 unit xrtl_net_URIUtils; //ExtendedRTL
22338: 228 unit uPSI_uft; (FFT) //DMath
22339: 229 unit uPSI_DBXChannel; //Delphi
22340: 230 unit uPSI_DBXIndyChannel; //Delphi Indy
22341: 231 unit uPSI_xrtl_util_COMCat; //ExtendedRTL
22342: 232 unit uPSI_xrtl_util_StrUtils; //ExtendedRTL
22343: 233 unit uPSI_xrtl_util_VariantUtils; //ExtendedRTL
22344: 234 unit uPSI_xrtl_util_FileUtils; //ExtendedRTL
22345: 235 unit xrtl_util_Compat; //ExtendedRTL
22346: 236 unit uPSI_OleAuto; //Borland
22347: 237 unit uPSI_xrtl_util_COMUtils; //ExtendedRTL
22348: 238 unit uPSI_CmAdmCtl; //Borland
22349: 239 unit uPSI_ValEdit2; //VCL
22350: 240 unit uPSI_GR32; //Graphics32
22351: 241 unit uPSI_GR32_Image; //Graphics32
22352: 242 unit uPSI_xrtl_util_TimeUtils; //ExtendedRTL
22353: 243 unit uPSI_xrtl_util_TimeZone; //ExtendedRTL
22354: 244 unit uPSI_xrtl_util_TimeStamp; //ExtendedRTL
22355: 245 unit uPSI_xrtl_util_Map; //ExtendedRTL
22356: 246 unit uPSI_xrtl_util_Set; //ExtendedRTL
22357: 247 unit uPSI_CPortMonitor; //ComPort
22358: 248 unit uPSI_StInistm; //SysTools4
22359: 249 unit uPSI_GR32_ExtImage; //Graphics32
22360: 250 unit uPSI_GR32_OrdinalMaps; //Graphics32

```

```

22361: 251 unit uPSI_GR32_Rasterizers; //Graphics32
22362: 252 unit uPSI_xrtl_util_Exception; //ExtendedRTL
22363: 253 unit uPSI_xrtl_util_Value; //ExtendedRTL
22364: 254 unit uPSI_xrtl_util_Compare; //ExtendedRTL
22365: 255 unit uPSI_FlatSB; //VCL
22366: 256 unit uPSI_JvAnalogClock; //JCL
22367: 257 unit uPSI_JvAlarms; //JCL
22368: 258 unit uPSI_JvSQLS; //JCL
22369: 259 unit uPSI_JvDBSecur; //JCL
22370: 260 unit uPSI_JvDBQBE; //JCL
22371: 261 unit uPSI_JvStarfield; //JCL
22372: 262 unit uPSI_JvCLMiscal; //JCL
22373: 263 unit uPSI_JvProfiler32; //JCL
22374: 264 unit uPSI_JvDirectories; //JCL
22375: 265 unit uPSI_JclSchedule; //JCL
22376: 266 unit uPSI_JclSvcCtrl; //JCL
22377: 267 unit uPSI_JvSoundControl; //JCL
22378: 268 unit uPSI_JvBDESQLScript; //JCL
22379: 269 unit uPSI_JvgDigits; //JCL>
22380: 270 unit uPSI_ImgList; //TCustomImageList
22381: 271 unit uPSI_JclMIDI; //JCL>
22382: 272 unit uPSI_JclWinMidi; //JCL>
22383: 273 unit uPSI_JclNTFS; //JCL>
22384: 274 unit uPSI_JclAppInst; //JCL>
22385: 275 unit uPSI_JvRle; //JCL>
22386: 276 unit uPSI_JvRas32; //JCL>
22387: 277 unit uPSI_JvImageDrawThread; //JCL>
22388: 278 unit uPSI_JvImageWindow; //JCL>
22389: 279 unit uPSI_JvTransparentForm; //JCL>
22390: 280 unit uPSI_JvWinDialogs; //JCL>
22391: 281 unit uPSI_JvSimLogic; //JCL>
22392: 282 unit uPSI_JvSimIndicator; //JCL>
22393: 283 unit uPSI_JvSimPID; //JCL>
22394: 284 unit uPSI_JvSimPIDLinker; //JCL>
22395: 285 unit uPSI_IdRFCReply; //Indy
22396: 286 unit uPSI_IdIdent; //Indy
22397: 287 unit uPSI_IdIdentServer; //Indy
22398: 288 unit uPSI_JvPatchFile; //JCL
22399: 289 unit uPSI_StNetPfm; //SysTools4
22400: 290 unit uPSI_StNet; //SysTools4
22401: 291 unit uPSI_JclPelage; //JCL
22402: 292 unit uPSI_JclPrint; //JCL
22403: 293 unit uPSI_JclMime; //JCL
22404: 294 unit uPSI_JvRichEdit; //JCL
22405: 295 unit uPSI_JvDBRichEd; //JCL
22406: 296 unit uPSI_JvDice; //JCL
22407: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
22408: 298 unit uPSI_JvDirFrm; //JCL
22409: 299 unit uPSI_JvDualList; //JCL
22410: 300 unit uPSI_JvSwitch; //JCL
22411: 301 unit uPSI_JvTimerLst; //JCL
22412: 302 unit uPSI_JvMemTable; //JCL
22413: 303 unit uPSI_JvObjStr; //JCL
22414: 304 unit uPSI_StLArr; //SysTools4
22415: 305 unit uPSI_StWmDCpy; //SysTools4
22416: 306 unit uPSI_StText; //SysTools4
22417: 307 unit uPSI_StNTLog; //SysTools4
22418: 308 unit uPSI_xrtl_math_Integer; //ExtendedRTL
22419: 309 unit uPSI_JvImagPrvw; //JCL
22420: 310 unit uPSI_JvFormPatch; //JCL
22421: 311 unit uPSI_JvPicClip; //JCL
22422: 312 unit uPSI_JvDataConv; //JCL
22423: 313 unit uPSI_JvCpuUsage; //JCL
22424: 314 unit uPSI_JclUnitConv_mx2; //JCL
22425: 315 unit JvDualListForm; //JCL
22426: 316 unit uPSI_JvCpuUsage2; //JCL
22427: 317 unit uPSI_JvParserForm; //JCL
22428: 318 unit uPSI_JvJanTreeView; //JCL
22429: 319 unit uPSI_JvTransLED; //JCL
22430: 320 unit uPSI_JvPlaylist; //JCL
22431: 321 unit uPSI_JvFormAutoSize; //JCL
22432: 322 unit uPSI_JvYearGridEditForm; //JCL
22433: 323 unit uPSI_JvMarkupCommon; //JCL
22434: 324 unit uPSI_JvChart; //JCL
22435: 325 unit uPSI_JvXPCore; //JCL
22436: 326 unit uPSI_JvXPCoreUtils; //JCL
22437: 327 unit uPSI_StatsClasses; //mx4
22438: 328 unit uPSI_ExtCtrls2; //VCL
22439: 329 unit uPSI_JvUrlGrabbers; //JCL
22440: 330 unit uPSI_JvXmlTree; //JCL
22441: 331 unit uPSI_JvWavePlayer; //JCL
22442: 332 unit uPSI_JvUnicodeCanvas; //JCL
22443: 333 unit uPSI_JvTFUUtils; //JCL
22444: 334 unit uPSI_IdServerIOHandler; //Indy
22445: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
22446: 336 unit uPSI_IdMessageCoder; //Indy
22447: 337 unit uPSI_IdMessageCoderMIME; //Indy
22448: 338 unit uPSI_IdMIMETypes; //Indy
22449: 339 unit uPSI_JvConverter; //JCL

```

```

22450: 340 unit uPSI_JvCsvParse; //JCL
22451: 341 unit uPSI_umath; unit uPSI_ugamma; //DMath
22452: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
22453: 343 unit uPSI_JvDBGridExport; //JCL
22454: 344 unit uPSI_JvgExport; //JCL
22455: 345 unit uPSI_JvSerialMaker; //JCL
22456: 346 unit uPSI_JvWin32; //JCL
22457: 347 unit uPSI_JvPaintFX; //JCL
22458: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
22459: 349 unit uPSI_JvValidators; (preview) //JCL
22460: 350 unit uPSI_JvNTEventLog; //JCL
22461: 351 unit uPSI_ShellZipTool; //mX4
22462: 352 unit uPSI_JvJoystick; //JCL
22463: 353 unit uPSI_JvMailSlots; //JCL
22464: 354 unit uPSI_JclComplex; //JCL
22465: 355 unit uPSI_SynPdf; //Synopse
22466: 356 unit uPSI_Registry; //VCL
22467: 357 unit uPSI_TlHelp32; //VCL
22468: 358 unit uPSI_JclRegistry; //JCL
22469: 359 unit uPSI_JvAirBrush; //JCL
22470: 360 unit uPSI_mORMotReport; //Synopse
22471: 361 unit uPSI_JclLocales; //JCL
22472: 362 unit uPSI_SynEdit; //SynEdit
22473: 363 unit uPSI_SynEditTypes; //SynEdit
22474: 364 unit uPSI_SynMacroRecorder; //SynEdit
22475: 365 unit uPSI_LongIntList; //SynEdit
22476: 366 unit uPSI_devutils; //DevC
22477: 367 unit uPSI_SynEditMiscClasses; //SynEdit
22478: 368 unit uPSI_SynEditRegexSearch; //SynEdit
22479: 369 unit uPSI_SynEditHighlighter; //SynEdit
22480: 370 unit uPSI_SynHighlighterPas; //SynEdit
22481: 371 unit uPSI_JvSearchFiles; //JCL
22482: 372 unit uPSI_SynHighlighterAny; //Lazarus
22483: 373 unit uPSI_SynEditKeyCmds; //SynEdit
22484: 374 unit uPSI_SynEditMiscProcs; //SynEdit
22485: 375 unit uPSI_SynEditKbdHandler; //SynEdit
22486: 376 unit uPSI_JvAppInst; //JCL
22487: 377 unit uPSI_JvAppEvent; //JCL
22488: 378 unit uPSI_JvAppCommand; //JCL
22489: 379 unit uPSI_JvAnimTitle; //JCL
22490: 380 unit uPSI_JvAnimatedImage; //JCL
22491: 381 unit uPSI_SynEditExport; //SynEdit
22492: 382 unit uPSI_SynExportHTML; //SynEdit
22493: 383 unit uPSI_SynExportRTF; //SynEdit
22494: 384 unit uPSI_SynEditSearch; //SynEdit
22495: 385 unit uPSI_fMain_back; //maxbox;
22496: 386 unit uPSI_JvZoom; //JCL
22497: 387 unit uPSI_PMrand; //PM
22498: 388 unit uPSI_JvSticker; //JCL
22499: 389 unit uPSI_XmlVerySimple; //mX4
22500: 390 unit uPSI_Services; //ExtPascal
22501: 391 unit uPSI_ExtPascalUtils; //ExtPascal
22502: 392 unit uPSI_SocketsDelphi; //ExtPascal
22503: 393 unit uPSI_StBarC; //SysTools
22504: 394 unit uPSI_StDbBarC; //SysTools
22505: 395 unit uPSI_StBarPN; //SysTools
22506: 396 unit uPSI_StDbPNBC; //SysTools
22507: 397 unit uPSI_StDb2DBC; //SysTools
22508: 398 unit uPSI_StMoney; //SysTools
22509: 399 unit uPSI_JvForth; //JCL
22510: 400 unit uPSI_RestRequest; //mX4
22511: 401 unit uPSI_HttpRESTConnectionIndy; //mX4
22512: 402 unit uPSI_JvXmlDatabase; //update //JCL
22513: 403 unit uPSI_StAstro; //SysTools
22514: 404 unit uPSI_StSort; //SysTools
22515: 405 unit uPSI_StDate; //SysTools
22516: 406 unit uPSI_StDateSt; //SysTools
22517: 407 unit uPSI_StBase; //SysTools
22518: 408 unit uPSI_StVInfo; //SysTools
22519: 409 unit uPSI_JvBrowseFolder; //JCL
22520: 410 unit uPSI_JvBoxProcs; //JCL
22521: 411 unit uPSI_urandom; (unit uranuvag;) //DMath
22522: 412 unit uPSI_usimann; (unit ugenalg;) //DMath
22523: 413 unit uPSI_JvHighlighter; //JCL
22524: 414 unit uPSI_Diff; //mX4
22525: 415 unit uPSI_SpringWinAPI; //DSpring
22526: 416 unit uPSI_StBits; //SysTools
22527: 417 unit uPSI_TomDBQue; //mX4
22528: 418 unit uPSI_MultilangTranslator; //mX4
22529: 419 unit uPSI_HyperLabel; //mX4
22530: 420 unit uPSI_Starter; //mX4
22531: 421 unit uPSI_FileAssoc; //devC
22532: 422 unit uPSI_devFileMonitorX; //devC
22533: 423 unit uPSI_devrn; //devC
22534: 424 unit uPSI_devExec; //devC
22535: 425 unit uPSI_loysUtils; //devC
22536: 426 unit uPSI_DosCommand; //devC
22537: 427 unit uPSI_CppTokenizer; //devC
22538: 428 unit uPSI_JvHLParser; //devC

```

```

22539: 429 unit uPSI_JclMapi;                                //JCL
22540: 430 unit uPSI_JclShell;                               //JCL
22541: 431 unit uPSI_JclCOM;                                //JCL
22542: 432 unit uPSI_GR32_Math;                             //Graphics32
22543: 433 unit uPSI_GR32_LowLevel;                          //Graphics32
22544: 434 unit uPSI_SimpleHl;                              //mX4
22545: 435 unit uPSI_GR32_Filters;                           //Graphics32
22546: 436 unit uPSI_GR32_VectorMaps;                      //Graphics32
22547: 437 unit uPSI_cXMLFunctions;                         //Fundamentals 4
22548: 438 unit uPSI_JvTimer;                               //JCL
22549: 439 unit uPSI_cHTTPUtils;                            //Fundamentals 4
22550: 440 unit uPSI_cTLSUtils;                            //Fundamentals 4
22551: 441 unit uPSI_JclGraphics;                           //JCL
22552: 442 unit uPSI_JclSynch;                             //JCL
22553: 443 unit uPSI_IdTelnet;                            //Indy
22554: 444 unit uPSI_IdTelnetServer;                        //Indy
22555: 445 unit uPSI_IdEcho;                               //Indy
22556: 446 unit uPSI_IdEchoServer;                          //Indy
22557: 447 unit uPSI_IdEchoUDP;                            //Indy
22558: 448 unit uPSI_IdEchoUDPServer;                      //Indy
22559: 449 unit uPSI_IdSocks;                             //Indy
22560: 450 unit uPSI_IdAntiFreezeBase;                     //Indy
22561: 451 unit uPSI_IdHostnameServer;                      //Indy
22562: 452 unit uPSI_IdTunnelCommon;                        //Indy
22563: 453 unit uPSI_IdTunnelMaster;                        //Indy
22564: 454 unit uPSI_IdTunnelSlave;                         //Indy
22565: 455 unit uPSI_IdRSH;                               //Indy
22566: 456 unit uPSI_IdRSHServer;                          //Indy
22567: 457 unit uPSI_Spring_Cryptography_Utils;           //Spring4Delphi
22568: 458 unit uPSI_MapReader;                            //devC
22569: 459 unit uPSI_LibTar;                               //devC
22570: 460 unit uPSI_IdStack;                             //Indy
22571: 461 unit uPSI_IdBlockCipherIntercept;              //Indy
22572: 462 unit uPSI_IdChargenServer;                      //Indy
22573: 463 unit uPSI_IdFTPServer;                          //Indy
22574: 464 unit uPSI_IdException;                          //Indy
22575: 465 unit uPSI_utexplot;                            //DMath
22576: 466 unit uPSI_uwinstr;                            //DMath
22577: 467 unit uPSI_VarRecUtils;                          //devC
22578: 468 unit uPSI_JvStringListToHtml;                   //JCL
22579: 469 unit uPSI_JvStringHolder;                        //JCL
22580: 470 unit uPSI_IdCoder;                             //Indy
22581: 471 unit uPSI_SynHighlighterDfm;                   //Synedit
22582: 472 unit uHighlighterProcs; in 471                //Synedit
22583: 473 unit uPSI_LazFileUtils;                         //LCL
22584: 474 unit uPSI_IDECmdLine;                           //LCL
22585: 475 unit uPSI_lazMasks;                            //LCL
22586: 476 unit uPSI_ip_misc;                            //mX4
22587: 477 unit uPSI_Barcode;                            //LCL
22588: 478 unit uPSI_SimpleXML;                           //LCL
22589: 479 unit uPSI_JclIniFiles;                          //JCL
22590: 480 unit uPSI_D2XXUnit; { $X- }                   //FTDI
22591: 481 unit uPSI_JclDateTime;                          //JCL
22592: 482 unit uPSI_JclEDI;                             //JCL
22593: 483 unit uPSI_JclMiscel2;                          //JCL
22594: 484 unit uPSI_JclValidation;                       //JCL
22595: 485 unit uPSI_JclAnsiStrings; {-PString}          //JCL
22596: 486 unit uPSI_SynEditMiscProcs2;                  //Synedit
22597: 487 unit uPSI_JclStreams;                          //JCL
22598: 488 unit uPSI_QRCode;                            //mX4
22599: 489 unit uPSI_BlockSocket;                         //ExtPascal
22600: 490 unit uPSI_Masks_Utils;                         //VCL
22601: 491 unit uPSI_synautil;                           //Synapse!
22602: 492 unit uPSI_JclMath_Class;                        //JCL RTL
22603: 493 unit ugamdist; //Gamma function;             //DMath
22604: 494 unit uibeta, ucorrel; //IBeta;                //DMath
22605: 495 unit uPSI_SRMgr;                             //mX4
22606: 496 unit uPSI_HotLog;                            //mX4
22607: 497 unit uPSI_DebugBox;                           //mX4
22608: 498 unit uPSI_ustrings;                           //DMath
22609: 499 unit uPSI_uregtest;                           //DMath
22610: 500 unit uPSI_usimplex;                          //DMath
22611: 501 unit uPSI_uhyper;                            //DMath
22612: 502 unit uPSI_IdHL7;                            //Indy
22613: 503 unit uPSI_IdIPMCastBase;                    //Indy
22614: 504 unit uPSI_IdIPMCastServer;                  //Indy
22615: 505 unit uPSI_IdIPMCastClient;                  //Indy
22616: 506 unit uPSI_unlfit; //nlregression;            //DMath
22617: 507 unit uPSI_IdRawHeaders;                      //Indy
22618: 508 unit uPSI_IdRawClient;                        //Indy
22619: 509 unit uPSI_IdRawFunctions;                   //Indy
22620: 510 unit uPSI_IdTCPStream;                        //Indy
22621: 511 unit uPSI_IdSNPP;                            //Indy
22622: 512 unit uPSI_St2DBarC;                          //SysTools
22623: 513 unit uPSI_ImageWin; //FTL;                  //VCL
22624: 514 unit uPSI_CustomDrawTreeView; //FTL;          //VCL
22625: 515 unit uPSI_GraphWin; //FTL;                  //VCL
22626: 516 unit uPSI_actionMain; //FTL;                  //VCL
22627: 517 unit uPSI_StSpawn;                           //SysTools

```

```

22628: 518 unit uPSI_CtlPanel; //VCL
22629: 519 unit uPSI_IdLPR; //Indy
22630: 520 unit uPSI_SockRequestInterpreter; //Indy
22631: 521 unit uPSI_ulambert; //DMath
22632: 522 unit uPSI_ucholesk; //DMath
22633: 523 unit uPSI_SimpleDS; //VCL
22634: 524 unit uPSI_DBXSqlScanner; //VCL
22635: 525 unit uPSI_DBXMetaDataTable; //VCL
22636: 526 unit uPSI_Chart; //TEE
22637: 527 unit uPSI_TeeProcs; //TEE
22638: 528 unit mXBDEUtils; //mX4
22639: 529 unit uPSI_MDIEdit; //VCL
22640: 530 unit uPSI_CopyPsr; //VCL
22641: 531 unit uPSI_SockApp; //VCL
22642: 532 unit uPSI_AppEvnts; //VCL
22643: 533 unit uPSI_ExtActns; //VCL
22644: 534 unit uPSI_TeEngine; //TEE
22645: 535 unit uPSI_CoolMain; //browser //VCL
22646: 536 unit uPSI_StCRC; //SysTools
22647: 537 unit uPSI_StDecMth2; //SysTools
22648: 538 unit uPSI_frmExportMain; //Synedit
22649: 539 unit uPSI_SynDBEdit; //Synedit
22650: 540 unit uPSI_SynEditWildcardSearch; //Synedit
22651: 541 unit uPSI_BoldComUtils; //BOLD
22652: 542 unit uPSI_BoldIsoDateTime; //BOLD
22653: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
22654: 544 unit uPSI_BoldXMLRequests; //BOLD
22655: 545 unit uPSI_BoldStringList; //BOLD
22656: 546 unit uPSI_BoldfileHandler; //BOLD
22657: 547 unit uPSI_BoldContainers; //BOLD
22658: 548 unit uPSI_BoldQueryUserDlg; //BOLD
22659: 549 unit uPSI_BoldWinINet; //BOLD
22660: 550 unit uPSI_BoldQueue; //BOLD
22661: 551 unit uPSI_JvPcx; //JCL
22662: 552 unit uPSI_IdWhois; //Indy
22663: 553 unit uPSI_IdWhoisServer; //Indy
22664: 554 unit uPSI_IdGopher; //Indy
22665: 555 unit uPSI_IdDateTimeStamp; //Indy
22666: 556 unit uPSI_IdDayTimeServer; //Indy
22667: 557 unit uPSI_IdDayTimeUDP; //Indy
22668: 558 unit uPSI_IdDayTimeUDPServer; //Indy
22669: 559 unit uPSI_IdDICTServer; //Indy
22670: 560 unit uPSI_IdDiscardServer; //Indy
22671: 561 unit uPSI_IdDiscardUDPServer; //Indy
22672: 562 unit uPSI_IdMappedFTP; //Indy
22673: 563 unit uPSI_IdMappedPortTCP; //Indy
22674: 564 unit uPSI_IdGopherServer; //Indy
22675: 565 unit uPSI_IdQotdServer; //Indy
22676: 566 unit uPSI_JvRgbToHtml; //JCL
22677: 567 unit uPSI_JvRemLog; //JCL
22678: 568 unit uPSI_JvSysComp; //JCL
22679: 569 unit uPSI_JvTMTL; //JCL
22680: 570 unit uPSI_JvWinampAPI; //JCL
22681: 571 unit uPSI_MSysUtils; //mX4
22682: 572 unit uPSI_ESBMaths; //ESB
22683: 573 unit uPSI_ESBMaths2; //ESB
22684: 574 unit uPSI_uLkJSON; //Lk
22685: 575 unit uPSI_ZURL; //Zeos
22686: 576 unit uPSI_ZSysUtils; //Zeos
22687: 577 unit unaUtils internals //UNA
22688: 578 unit uPSI_ZMatchPattern; //Zeos
22689: 579 unit uPSI_ZClasses; //Zeos
22690: 580 unit uPSI_ZCollections; //Zeos
22691: 581 unit uPSI_ZEncoding; //Zeos
22692: 582 unit uPSI_IdRawBase; //Indy
22693: 583 unit uPSI_IdNTLM; //Indy
22694: 584 unit uPSI_IdNNTP; //Indy
22695: 585 unit uPSI_usniffer; //PortScanForm //mX4
22696: 586 unit uPSI_IdCoderMIME; //Indy
22697: 587 unit uPSI_IdCoderUUE; //Indy
22698: 588 unit uPSI_IdCoderXXE; //Indy
22699: 589 unit uPSI_IdCoder3to4; //Indy
22700: 590 unit uPSI_IdCookie; //Indy
22701: 591 unit uPSI_IdCookieManager; //Indy
22702: 592 unit uPSI_WDOSocketUtils; //WDOS
22703: 593 unit uPSI_WDOSPlcUtils; //WDOS
22704: 594 unit uPSI_WDOSPorts; //WDOS
22705: 595 unit uPSI_WDOSResolvers; //WDOS
22706: 596 unit uPSI_WDOSTimers; //WDOS
22707: 597 unit uPSI_WDOSPlcs; //WDOS
22708: 598 unit uPSI_WDOSPneumatics; //WDOS
22709: 599 unit uPSI_IdFingerServer; //Indy
22710: 600 unit uPSI_IdDDNSResolver; //Indy
22711: 601 unit uPSI_IdHTTPWebBrokerBridge; //Indy
22712: 602 unit uPSI_IdIntercept; //Indy
22713: 603 unit uPSI_IdIPMCastBase; //Indy
22714: 604 unit uPSI_IdLogBase; //Indy
22715: 605 unit uPSI_IdIOHandlerStream; //Indy
22716: 606 unit uPSI_IdMappedPortUDP; //Indy

```

```

22717: 607 unit uPSI_IdQOTDUDPServer; //Indy
22718: 608 unit uPSI_IdQOTDUDP; //Indy
22719: 609 unit uPSI_IdSysLog; //Indy
22720: 610 unit uPSI_IdSysLogServer; //Indy
22721: 611 unit uPSI_IdSysLogMessage; //Indy
22722: 612 unit uPSI_IdTimeServer; //Indy
22723: 613 unit uPSI_IdTimeUDP; //Indy
22724: 614 unit uPSI_IdTimeUDPServer; //Indy
22725: 615 unit uPSI_IdUserAccounts; //Indy
22726: 616 unit uPSI_TextUtils; //mX4
22727: 617 unit uPSI_MandelbrotEngine; //mX4
22728: 618 unit uPSI_delphi_arduino_Unit1; //mX4
22729: 619 unit uPSI_DTDSchema2; //mX4
22730: 620 unit uPSI_fpplotMain; //DMath
22731: 621 unit uPSI_FindFileIter; //mX4
22732: 622 unit uPSI_PppState; (JclStrHashMap) //PPP
22733: 623 unit uPSI_PppParser; //PPP
22734: 624 unit uPSI_PppLexer; //PPP
22735: 625 unit uPSI_PCharUtils; //PPP
22736: 626 unit uPSI_uJSON; //WU
22737: 627 unit uPSI_JclStrHashMap; //JCL
22738: 628 unit uPSI_JclHookExcept; //JCL
22739: 629 unit uPSI_EncdDecd; //VCL
22740: 630 unit uPSI_SockAppReg; //VCL
22741: 631 unit uPSI_PJFileHandle; //PJ
22742: 632 unit uPSI_PJEnvVars; //PJ
22743: 633 unit uPSI_PJPipe; //PJ
22744: 634 unit uPSI_PJPipeFilters; //PJ
22745: 635 unit uPSI_PJConsoleApp; //PJ
22746: 636 unit uPSI_UConsoleAppEx; //PJ
22747: 637 unit uPSI_DbxSocketChannelNative; //VCL
22748: 638 unit uPSI_DbxDataGenerator; //VCL
22749: 639 unit uPSI_DBXClient; //VCL
22750: 640 unit uPSI_IdLogEvent; //Indy
22751: 641 unit uPSI_Reversi; //mX4
22752: 642 unit uPSI_Geometry; //mX4
22753: 643 unit uPSI_IdSMTPServer; //Indy
22754: 644 unit uPSI_Textures; //mX4
22755: 645 unit uPSI_IBX; //VCL
22756: 646 unit uPSI_IWDBCommon; //VCL
22757: 647 unit uPSI_SortGrid; //mX4
22758: 648 unit uPSI_IB; //VCL
22759: 649 unit uPSI_IBScript; //VCL
22760: 650 unit uPSI_JvCSVBaseControls; //JCL
22761: 651 unit uPSI_Jvg3DColors; //JCL
22762: 652 unit uPSI_JvhLEditor; //beat //JCL
22763: 653 unit uPSI_JvShellHook; //JCL
22764: 654 unit uPSI_DBCommon2; //VCL
22765: 655 unit uPSI_JvSHFileOperation; //JCL
22766: 656 unit uPSI_uFilexport; //mX4
22767: 657 unit uPSI_JvDialogs; //JCL
22768: 658 unit uPSI_JvDBTreeView; //JCL
22769: 659 unit uPSI_JvDBUltimGrid; //JCL
22770: 660 unit uPSI_JvDBQueryParamsForm; //JCL
22771: 661 unit uPSI_JvExControls; //JCL
22772: 662 unit uPSI_JvBDEMemTable; //JCL
22773: 663 unit uPSI_JvCommStatus; //JCL
22774: 664 unit uPSI_JvMailSlots2; //JCL
22775: 665 unit uPSI_JvgWinMask; //JCL
22776: 666 unit uPSI_StEclipse; //SysTools
22777: 667 unit uPSI_StMime; //SysTools
22778: 668 unit uPSI_StList; //SysTools
22779: 669 unit uPSI_StMerge; //SysTools
22780: 670 unit uPSI_StStrS; //SysTools
22781: 671 unit uPSI_StTree; //SysTools
22782: 672 unit uPSI_StVArr; //SysTools
22783: 673 unit uPSI_StRegIni; //SysTools
22784: 674 unit uPSI_urkf; //DMath
22785: 675 unit uPSI_usvd; //DMath
22786: 676 unit uPSI_DepWalkUtils; //JCL
22787: 677 unit uPSI_OptionsFrm; //JCL
22788: 678 unit yuvconverts; //mX4
22789: 679 uPSI_JvPropAutoSave; //JCL
22790: 680 uPSI_AclAPI; //alcinoe
22791: 681 uPSI_AviCap; //alcinoe
22792: 682 uPSI_ALAVLBinaryTree; //alcinoe
22793: 683 uPSI_ALFcnnMisc; //alcinoe
22794: 684 uPSI_ALStringList; //alcinoe
22795: 685 uPSI_ALQuickSortList; //alcinoe
22796: 686 uPSI_ALStaticText; //alcinoe
22797: 687 uPSI_ALJSONDoc; //alcinoe
22798: 688 uPSI_ALGSMComm; //alcinoe
22799: 689 uPSI_ALWindows; //alcinoe
22800: 690 uPSI_ALMultiPartFormDataParser; //alcinoe
22801: 691 uPSI_ALHttpCommon; //alcinoe
22802: 692 uPSI_ALWebSpider; //alcinoe
22803: 693 uPSI_ALHttpClient; //alcinoe
22804: 694 uPSI_ALFcnnHTML; //alcinoe
22805: 695 uPSI_ALFTPClient; //alcinoe

```

```

22806: 696 uPSI_ALInternetMessageCommon; //alcinoe
22807: 697 uPSI_ALWininetHttpClient; //alcinoe
22808: 698 uPSI_ALWinInetFTPCClient; //alcinoe
22809: 699 uPSI_ALWinHttpWrapper; //alcinoe
22810: 700 uPSI_ALWinHttpClient; //alcinoe
22811: 701 uPSI_ALFcnsWinSock; //alcinoe
22812: 702 uPSI_ALFcnsSQL; //alcinoe
22813: 703 uPSI_ALFcnsCGI; //alcinoe
22814: 704 uPSI_ALFcnsExecute; //alcinoe
22815: 705 uPSI_ALFcnsFile; //alcinoe
22816: 706 uPSI_ALFcnsMime; //alcinoe
22817: 707 uPSI_ALPhpRunner; //alcinoe
22818: 708 uPSI_ALGraphic; //alcinoe
22819: 709 uPSI_ALIniFiles; //alcinoe
22820: 710 uPSI_ALMemCachedClient; //alcinoe
22821: 711 unit uPSI_MyGrids; //mX4
22822: 712 uPSI_ALMultiPartMixedParser; //alcinoe
22823: 713 uPSI_ALSMTPClient; //alcinoe
22824: 714 uPSI_ALNNTPClient; //alcinoe
22825: 715 uPSI_ALHintBalloon; //alcinoe
22826: 716 unit uPSI_ALXmlDoc; //alcinoe
22827: 717 unit uPSI_IPCThrd; //VCL
22828: 718 unit uPSI_MonForm; //VCL
22829: 719 unit uPSI_TeCanvas; //Orpheus
22830: 720 unit uPSI_OvcMisc; //Orpheus
22831: 721 unit uPSI_ovcfiler; //Orpheus
22832: 722 unit uPSI_ovcstate; //Orpheus
22833: 723 unit uPSI_ovccoco; //Orpheus
22834: 724 unit uPSI_ovcrvexp; //Orpheus
22835: 725 unit uPSI_OvcFormatSettings; //Orpheus
22836: 726 unit uPSI_OvcUtils; //Orpheus
22837: 727 unit uPSI_ovcstore; //Orpheus
22838: 728 unit uPSI_ovcstr; //Orpheus
22839: 729 unit uPSI_ovcmru; //Orpheus
22840: 730 unit uPSI_ovccmd; //Orpheus
22841: 731 unit uPSI_ovctimer; //Orpheus
22842: 732 unit uPSI_ovcintl; //Orpheus
22843: 733 uPSI_AfCircularBuffer; //AsyncFree
22844: 734 uPSI_AfUtils; //AsyncFree
22845: 735 uPSI_AfSafeSync; //AsyncFree
22846: 736 uPSI_AfComPortCore; //AsyncFree
22847: 737 uPSI_AfComPort; //AsyncFree
22848: 738 uPSI_AfPortControls; //AsyncFree
22849: 739 uPSI_AfDataDispatcher; //AsyncFree
22850: 740 uPSI_AfViewers; //AsyncFree
22851: 741 uPSI_AfDataTerminal; //AsyncFree
22852: 742 uPSI_SimplePortMain; //AsyncFree
22853: 743 unit uPSI_ovcclock; //Orpheus
22854: 744 unit uPSI_o32intlst; //Orpheus
22855: 745 unit uPSI_o32ledlabel; //Orpheus
22856: 746 unit uPSI_AlMySqlClient; //alcinoe
22857: 747 unit uPSI_ALFBXClient; //alcinoe
22858: 748 unit uPSI_ALFcnsSQL; //alcinoe
22859: 749 unit uPSI_AsyncTimer; //mX4
22860: 750 unit uPSI_ApplicationFileIO; //mX4
22861: 751 unit uPSI_PsAPI; //VCLé
22862: 752 uPSI_ovcuser; //Orpheus
22863: 753 uPSI_ovcurl; //Orpheus
22864: 754 uPSI_ovcvlb; //Orpheus
22865: 755 uPSI_ovccolor; //Orpheus
22866: 756 uPSI_ALFBXLib; //alcinoe
22867: 757 uPSI_ovcmeter; //Orpheus
22868: 758 uPSI_ovcpeakm; //Orpheus
22869: 759 uPSI_o32BGSty; //Orpheus
22870: 760 uPSI_ovcBidi; //Orpheus
22871: 761 uPSI_ovctcarry; //Orpheus
22872: 762 uPSI_DXPUtilities; //mX4
22873: 763 uPSI_ALMultiPartBaseParser; //alcinoe
22874: 764 uPSI_ALMultiPartAlternativeParser; //alcinoe
22875: 765 uPSI_ALPOP3Client; //alcinoe
22876: 766 uPSI_SmallUtils; //mX4
22877: 767 uPSI_MakeApp; //mX4
22878: 768 uPSI_o32MouseMon; //Orpheus
22879: 769 uPSI_OvcCache; //Orpheus
22880: 770 uPSI_ovccalc; //Orpheus
22881: 771 uPSI_Joystick; //OpenGL
22882: 772 uPSI_ScreenSaver; //OpenGL
22883: 773 uPSI_XCollection; //OpenGL
22884: 774 uPSI_Polynomials; //OpenGL
22885: 775 uPSI_PersistentClasses, //9.86 //OpenGL
22886: 776 uPSI_VectorLists; //OpenGL
22887: 777 uPSI_XOpenGL; //OpenGL
22888: 778 uPSI_MeshUtils; //OpenGL
22889: 779 unit uPSI_JclSysUtils; //JCL
22890: 780 unit uPSI_JclBorlandTools; //JCL
22891: 781 unit JclFileUtils_max; //JCL
22892: 782 uPSI_AfDataControls; //AsyncFree
22893: 783 uPSI_GLSilhouette; //OpenGL
22894: 784 uPSI_JclSysUtils_class; //JCL

```

```

22895: 785 uPSI_JclFileUtils_class;           //JCL
22896: 786 uPSI_FileUtil;                   //JCL
22897: 787 uPSI_changefind;                //mX4
22898: 788 uPSI_CmdIntf;                  //mX4
22899: 789 uPSI_fservice;                 //mX4
22900: 790 uPSI_Keyboard;                 //OpenGL
22901: 791 uPSI_VRMLParser;               //OpenGL
22902: 792 uPSI_GLFileVRML;              //OpenGL
22903: 793 uPSI_Octree;                  //OpenGL
22904: 794 uPSI_GLPolyhedron;             //OpenGL
22905: 795 uPSI_GLCrossPlatform;          //OpenGL
22906: 796 uPSI_GLParticles;              //OpenGL
22907: 797 uPSI_GLNavigator;              //OpenGL
22908: 798 uPSI_GLStarRecord;             //OpenGL
22909: 799 uPSI_GLTextureCombiners;       //OpenGL
22910: 800 uPSI_GLCanvas;                 //OpenGL
22911: 801 uPSI_GeometryBB;               //OpenGL
22912: 802 uPSI_GeometryCoordinates;      //OpenGL
22913: 803 uPSI_VectorGeometry;            //OpenGL
22914: 804 uPSI_BumpMapping;              //OpenGL
22915: 805 uPSI_TGA;                     //OpenGL
22916: 806 uPSI_GLVectorFileObjects;      //OpenGL
22917: 807 uPSI_IMM;                     //VCL
22918: 808 uPSI_CategoryButtons;          //VCL
22919: 809 uPSI_ButtonGroup;              //VCL
22920: 810 uPSI_DbExcept;                //VCL
22921: 811 uPSI_AxCtrls;                 //VCL
22922: 812 uPSI_GL_actorUnit1;            //OpenGL
22923: 813 uPSI_StdVCL;                  //VCL
22924: 814 unit CurvesAndSurfaces;        //OpenGL
22925: 815 uPSI_DataAwareMain;             //AsyncFree
22926: 816 uPSI_TabNotBk;                 //VCL
22927: 817 uPSI_udwsfiler;                //mX4
22928: 818 uPSI_synaip;                  //Synapse!
22929: 819 uPSI_synacode;                //Synapse
22930: 820 uPSI_synachar;                //Synapse
22931: 821 uPSI_synamisc;                //Synapse
22932: 822 uPSI_synaser;                 //Synapse
22933: 823 uPSI_synaicnv;                //Synapse
22934: 824 uPSI_tlnitsend;                //Synapse
22935: 825 uPSI_pingsend;                //Synapse
22936: 826 uPSI_blksock;                 //Synapse
22937: 827 uPSI_asnutil;                 //Synapse
22938: 828 uPSI_dnssend;                 //Synapse
22939: 829 uPSI_clamsend;                //Synapse
22940: 830 uPSI_ldapsend;                //Synapse
22941: 831 uPSI_mimemess;                //Synapse
22942: 832 uPSI_slogsend;                //Synapse
22943: 833 uPSI_mimepart;                //Synapse
22944: 834 uPSI_mimeinln;                //Synapse
22945: 835 uPSI_ftpsend;                //Synapse
22946: 836 uPSI_ftptsend;                //Synapse
22947: 837 uPSI_httpsend;                //Synapse
22948: 838 uPSI_sntpsend;                //Synapse
22949: 839 uPSI_smtpsend;                //Synapse
22950: 840 uPSI_snmpsend;                //Synapse
22951: 841 uPSI_imapsend;                //Synapse
22952: 842 uPSI_pop3send;                //Synapse
22953: 843 uPSI_nntpsend;                //Synapse
22954: 844 uPSI_ssl_cryptlib;             //Synapse
22955: 845 uPSI_ssl_openssl;              //Synapse
22956: 846 uPSI_synhttp_daemon;           //Synapse
22957: 847 uPSI_NetWork;                 //mX4
22958: 848 uPSI_PingThread;               //Synapse
22959: 849 uPSI_JvThreadTimer;             //JCL
22960: 850 unit uPSI_wwSystem;             //InfoPower
22961: 851 unit uPSI_IdComponent;          //Indy
22962: 852 unit uPSI_IdIOHandlerThrottle; //Indy
22963: 853 unit uPSI_Themes;               //VCL
22964: 854 unit uPSI_StdStyleActnCtrls;    //VCL
22965: 855 unit uPSI_UDDIHelper;            //VCL
22966: 856 unit uPSI_IdIMAP4Server;        //Indy
22967: 857 uPSI_VariantSymbolTable;         //VCL //3.9.9.92
22968: 858 uPSI_udf_glob;                 //mX4
22969: 859 uPSI_TabGrid;                  //VCL
22970: 860 uPSI_JsDBTreeView;              //mX4
22971: 861 uPSI_JsSendMail;                //mX4
22972: 862 uPSI_dbTvRecordList;             //mX4
22973: 863 uPSI_TreeVwEx;                 //mX4
22974: 864 uPSI_ECDataLink;                //mX4
22975: 865 uPSI_dbTree;                   //mX4
22976: 866 uPSI_dbTreeCBox;                //mX4
22977: 867 unit uPSI_Debug; //TfrmDebug //mX4
22978: 868 uPSI_TimeFncts;                //mX4
22979: 869 uPSI_FileIntf;                 //VCL
22980: 870 uPSI_SockTransport;              //RTL
22981: 871 unit uPSI_WinInet;               //RTL
22982: 872 unit uPSI_Wwstr;                 //mX4
22983: 873 uPSI_DBLookup;                  //VCL

```

```

22984: 874 uPSI_Hotspot, //mX4
22985: 875 uPSI_HList; //History List //mX4
22986: 876 unit uPSI_DrTable; //VCL
22987: 877 uPSI_TConnect, //VCL
22988: 878 uPSI_DataBkr, //VCL
22989: 879 uPSI_HTTPIntr; //VCL
22990: 880 unit uPSI_Mathbox; //mX4
22991: 881 uPSI_cyIndy, //cY
22992: 882 uPSI_cySysUtils, //cY
22993: 883 uPSI_cyWinUtils, //cY
22994: 884 uPSI_cyStrUtils, //cY
22995: 885 uPSI_cyObjUtils, //cY
22996: 886 uPSI_cyDateUtils, //cY
22997: 887 uPSI_cyBDE, //cY
22998: 888 uPSI_cyClasses, //cY
22999: 889 uPSI_cyGraphics, //3.9.9.94_2 //cY
23000: 890 unit uPSI_cyTypes; //cY
23001: 891 uPSI_JvDatePicker, //JCL
23002: 892 uPSI_JvCreateProcess, //JCL
23003: 893 uPSI_JvEasterEgg, //JCL
23004: 894 uPSI_WinSvc, //3.9.9.94_3 //VCL
23005: 895 uPSI_SvcMgr //VCL
23006: 896 unit uPSI_JvPickDate; //JCL
23007: 897 unit uPSI_JvNotify; //JCL
23008: 898 uPSI_JvStrHlder //JCL
23009: 899 unit uPSI_JclNTFS2; //JCL
23010: 900 uPSI_Jcl8087 //math coprocessor //JCL
23011: 901 uPSI_JvAddPrinter //JCL
23012: 902 uPSI_JvCabFile //JCL
23013: 903 uPSI_JvDataEmbedded; //JCL
23014: 904 unit uPSI_U_HexView; //mX4
23015: 905 uPSI_UWavein4, //mX4
23016: 906 uPSI_AMixer, //mX4
23017: 907 uPSI_JvaScrollText, //mX4
23018: 908 uPSI_JvArrow, //mX4
23019: 909 unit uPSI.UrlMon; //mX4
23020: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas' //mX4
23021: 911 unit uPSI_U_Oscilloscope4; //ToscfrmMain; //DFF
23022: 912 unit uPSI_DFFUtils; //DFF
23023: 913 unit uPSI_MathsLib; //DFF
23024: 914 uPSI_UIntlist; //DFF
23025: 915 uPSI_UGetParens; //DFF
23026: 916 unit uPSI_UGeometry; //DFF
23027: 917 unit uPSI_UAstronomy; //DFF
23028: 918 unit uPSI_UCardComponentV2; //DFF
23029: 919 unit uPSI_UTGraphSearch; //DFF
23030: 920 unit uPSI_UParser10; //DFF
23031: 921 unit uPSI_cyIEUtils; //cY
23032: 922 unit uPSI_UcomboV2; //DFF
23033: 923 uPSI_cyBaseComm, //cY
23034: 924 uPSI_cyAppInstances, //cY
23035: 925 uPSI_cyAttract, //cY
23036: 926 uPSI_cyDERUtils //cY
23037: 927 unit uPSI_cyDocER; //cY
23038: 928 unit uPSI_ODBC; //mX
23039: 929 unit uPSI_AssocExec; //mX
23040: 930 uPSI_cyBaseCommRoomConnector, //cY
23041: 931 uPSI_cyCommRoomConnector, //cY
23042: 932 uPSI_cyCommunicate, //cY
23043: 933 uPSI_cyImage; //cY
23044: 934 uPSI_cyBaseContainer //cY
23045: 935 uPSI_cyModalContainer, //cY
23046: 936 uPSI_cyFlyingContainer; //cY
23047: 937 uPSI_RegStr, //VCL
23048: 938 uPSI_HtmlHelpViewer; //VCL
23049: 939 unit uPSI_cyInifrom //cY
23050: 940 unit uPSI_cyVirtualGrid; //cY
23051: 941 uPSI_Profiler, //DA
23052: 942 uPSI_BackgroundWorker, //DA
23053: 943 uPSI_Waveplay, //DA
23054: 944 uPSI_WaveTimer, //DA
23055: 945 uPSI_Waveutils; //DA
23056: 946 uPSI_NamedPipes, //TB
23057: 947 uPSI_NamedPipeServer, //TB
23058: 948 unit uPSI_Process, //TB
23059: 949 unit uPSI_DPUtills; //TB
23060: 950 unit uPSI_CommonTools; //TB
23061: 951 uPSI_DataSendToWeb, //TB
23062: 952 uPSI_StarCalc, //TB
23063: 953 uPSI_D2_XPVistaHelperU //TB
23064: 954 unit uPSI_NetTools //TB
23065: 955 unit uPSI_Pipes //TB
23066: 956 uPSI_ProcessUnit, //mX
23067: 957 uPSI_adGSM, //mX
23068: 958 unit uPSI_BetterADODataset; //mX
23069: 959 unit uPSI_AdSelCom; //FTT //mX
23070: 960 unit unit uPSI_dwsXplatform; //DWS
23071: 961 uPSI_AdSocket; //mX Turbo Power
23072: 962 uPSI_AdPacket; //mX

```

```

23073: 963 uPSI_AdPort; //mX
23074: 964 uPSI_PathFunc; //Inno
23075: 965 uPSI_CmnFunc; //Inno
23076: 966 uPSI_CmnFunc2; //Inno Setup //Inno
23077: 967 unit uPSI_BitmapImage; //mX4
23078: 968 unit uPSI_ImageGrabber; //mX4
23079: 969 uPSI_SecurityFunc; //Inno
23080: 970 uPSI_RedirFunc; //Inno
23081: 971 uPSI_FIFO, (MemoryStream) //mX4
23082: 972 uPSI_Int64Em; //Inno
23083: 973 unit uPSI_InstFunc; //Inno
23084: 974 unit uPSI_LibFusion; //Inno
23085: 975 uPSI_SimpleExpression; //Inno
23086: 976 uPSI_unitResourceDetails, //XN
23087: 977 uPSI_unitResFile, //XN
23088: 978 unit uPSI_simpleComport; //mX4
23089: 979 unit uPSI_AfViewershelpers; //Async
23090: 980 unit uPSI_Console; //mX4
23091: 981 unit uPSI_AnalogMeter; //TB
23092: 982 unit uPSI_XPrinter; //TB
23093: 983 unit uPSI_IniFiles; //VCL
23094: 984 unit uPSI_lazIniFiles; //FP
23095: 985 uPSI_testutils; //FP
23096: 986 uPSI_ToolsUnit; (DBTests) //FP
23097: 987 uPSI_fpcunit //FP
23098: 988 uPSI_testdecorator; //FP
23099: 989 unit uPSI_fpcunittests; //FP
23100: 990 unit uPSI_cTCPBuffer; //Fundamentals 4
23101: 991 unit uPSI_Glut; //Open GL
23102: 992 uPSI_LEDBitmaps, //mX4
23103: 993 uPSI_FileClass; //Inno
23104: 994 uPSI_FileUtilsClass; //mX4
23105: 995 uPSI_ComPortInterface; //Kit //mX4
23106: 996 unit uPSI_SwitchLed; //mX4
23107: 997 unit uPSI_cyDmmCanvas; //cY
23108: 998 uPSI_uColorFunctions; //DFF
23109: 999 uPSI_uSettings; //DFD
23110: 1000 uPSI_cyDebug.pas //cY
23111: 1001 uPSI_cyColorMatrix; //cY
23112: 1002 unit uPSI_cyCopyFiles; //cY
23113: 1003 unit uPSI_cySearchfiles; //cY
23114: 1004 unit uPSI_cyBaseMeasure; //cY
23115: 1005 unit uPSI_PJIStreams; //DD
23116: 1006 unit uPSI_cyRunTimeResize; //cY
23117: 1007 unit uPSI_jcontrolutils; //cY
23118: 1008 unit uPSI_kcMapViewer; (+GEONames) //kc
23119: 1009 unit uPSI_kcMapViewerDESynapse; //kc
23120: 1010 unit uPSI_cparserutils; (+GIS_SysUtils) //kc
23121: 1011 unit uPSI_LedNumber; //TurboPower
23122: 1012 unit uPSI_StStrL; //SysTools
23123: 1013 unit uPSI_indGhouMeter; //LAZ
23124: 1014 unit uPSI_Sensors; //LAZ
23125: 1015 unit uPSI_pwmain; //cgi of powtils //Pow
23126: 1016 unit uPSI_HTMLUtil; //Pow
23127: 1017 unit uPSI_synwrap1; //httpsend //Pow
23128: 1018 unit StreamWrap1 //Pow
23129: 1019 unit uPSI_pwmain; //Pow
23130: 1020 unit pwtypes //Pow
23131: 1021 uPSI_W32VersionInfo //LAZ
23132: 1022 unit uPSI_IpAnim; //LAZ
23133: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
23134: 1024 unit uPSI_LrtPoTools; //LAZ
23135: 1025 unit uPSI_Laz_DOM; //LAZ
23136: 1026 unit uPSI_hhAvComp; //LAZ
23137: 1027 unit uPSI_GPS2; //mX4
23138: 1028 unit uPSI_GPS; //mX4
23139: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo //mX4
23140: 1030 unit uPSI_NMEA; // GPS //mX4
23141: 1031 unit uPSI_ScreenThreeDLab; //mX4
23142: 1032 unit uPSI_Spin; //VCL
23143: 1033 unit uPSI_DynaZip; //mX4
23144: 1034 unit uPSI_clockExpert; //TB
23145: 1035 unit debugLn //mX4
23146: 1036 uPSI_SortUtils; //Jcl
23147: 1037 uPSI_BitmapConversion; //Jcl
23148: 1038 unit uPSI_JclTD32; //Jcl
23149: 1039 unit uPSI_ZDbcUtils; //Zeos
23150: 1040 unit uPSI_ZScriptParser; //Zeos
23151: 1041 uPSI_JvIni; //JCL
23152: 1042 uPSI_JvFtpGrabber; //JCL
23153: 1043 unit uPSI_NeuralNetwork; //OCL
23154: 1044 unit uPSI_StExpr; //SysTools
23155: 1045 unit uPSI_GR32_Geometry; //GR32
23156: 1046 unit uPSI_GR32_Containers; //GR32
23157: 1047 unit uPSI_GR32_Backends_VCL; //GR32
23158: 1048 unit uPSI_StSaturn; //Venus+Mercury+Mars++ //SysTools
23159: 1049 unit uPSI_JclParseUses; //JCL
23160: 1050 unit uPSI_JvFinalize; //JCL
23161: 1051 unit uPSI_panUnit1; //GLScene

```

```

23162: 1052 unit uPSI_DD83ul; //Arduino Tester          //mX4
23163: 1053 unit uPSI_BigIni                         //Hinzen
23164: 1054 unit uPSI_ShellCtrls;                   //VCL
23165: 1055 unit uPSI_fmath;                        //FMath
23166: 1056 unit uPSI_fComp;                       //FMath
23167: 1057 unit uPSI_HighResTimer;                //Lauer
23168: 1058 unit uconvMain; (Unit Converter)        //PS
23169: 1059 unit uPSI_uconvMain;                   //PS
23170: 1060 unit uPSI_ParserUtils;                 //PS
23171: 1061 unit uPSI_UPSUtils;                   //PS
23172: 1062 unit uPSI_ParserU;                    //PS
23173: 1063 unit uPSI_TypeInfo;                   //VCL
23174: 1064 unit uPSI_ServiceMgr;                 //mX
23175: 1065 unit uPSI_UDict;                      //DFP
23176: 1066 unit uPSI_ubigFloatV3;                //DFP
23177: 1067 unit uPSI_UBigIntsV4;                 //DFP
23178: 1068 unit uPSI_ServiceMgr2;                //mX
23179: 1069 unit uPSI_UP10Build;                  //PS
23180: 1070 unit uPSI_UParser10;                  //PS
23181: 1071 unit uPSI_IdModBusServer;             //MB
23182: 1072 unit uPSI_IdModBusClient;             //MB
23183: 1073 unit uPSI_ColorGrd;                  //VCL
23184: 1074 unit uPSI_DirOutln;                  //VCL
23185: 1075 unit uPSI_Gauges;                    //VCL
23186: 1076 unit uPSI_CustomizeDlg;              //VCL
23187: 1077 unit uPSI_ActnMan;                  //VCL
23188: 1078 unit uPSI_CollPanl;                  //VCL
23189: 1079 unit uPSI_Calendar2;                //VCL
23190: 1080 unit uPSI_IBCtrls;                  //VCL
23191: 1081 unit uPSI_IdStackWindows;            //Indy
23192: 1082 unit uPSI_CTSVendorUtils;            //DBX
23193: 1083 unit uPSI_VendorTestFramework;       //DBX
23194: 1084 unit uPSI_TInterval;                //mX4
23195: 1085 unit uPSI_JvAnimate;                //JCL
23196: 1086 unit uPSI_DBXCharDecoder;            //DBX
23197: 1087 unit uPSI_JvDBLists;                //JCL
23198: 1088 unit uPSI_JvFileInfo;               //JCL
23199: 1089 unit uPSI_SOAPConn;                 //VCL
23200: 1090 unit uPSI_SOAPLinked;              //VCL
23201: 1091 unit uPSI_XSBuiltIns;              //VCL
23202: 1092 unit uPSI_JvgDigits;               //JCL
23203: 1093 unit uPSI_JvDesignUtils;            //JCL
23204: 1094 unit uPSI_JvgCrossTable;           //JCL
23205: 1095 unit uPSI_JvgReport;               //JCL
23206: 1096 unit uPSI_JvDBRichEdit;             //JCL
23207: 1097 unit uPSI_JvWinHelp;                //JCL
23208: 1098 unit uPSI_WaveConverter;            //JCL
23209: 1099 unit uPSI_ACMConvertor;             //JCL
23210: 1100 unit XSBuiltIns_Routines;          //JCL
23211: 1101 unit uPSI_ComObjOleDB_utils.pas;   //JCL
23212: 1102 unit uPSI_SMScript;                //JCL
23213: 1103 unit uPSI_CompFileIo;              //JCL
23214: 1104 unit uPSI_SynHighlighterGeneral;    //JCL
23215: 1105 unit uPSI_geometry2;               //JCL
23216: 1106 unit uPSI_MConnect;                //JCL
23217: 1107 unit uPSI_ObjBrkr;                 //JCL
23218: 1108 unit uPSI_uMultiStr;               //JCL
23219: 1109 unit uPSI_WinAPI.pas;              //JCL
23220: 1110 unit uPSI_JvAVICapture;            //JCL
23221: 1111 unit uPSI_JvExceptionForm;         //JCL
23222: 1112 unit uPSI_JvConnectNetwork;         //JCL
23223: 1113 unit uPSI_MMMainForm;              //JCL
23224: 1114 unit uPSI_DdeMan;                 //JCL
23225: 1115 unit uPSI_DIUtilities;             //JCL
23226: 1116 unit uPSI_gnugettext;              //JCL
23227: 1117 unit uPSI_Xmlxform;               //JCL
23228: 1118 unit uPSI_SvrHTTPIndy;             //JCL
23229: 1119 unit uPSI_CPortTrmSet;             //JCL
23230: 1120 unit SvrLog;                      //JCL
23231:
23232:
23233:
23234: http://www.slideshare.net/maxkleiner1/codereview-topics
23235: /////////////////////////////////
23236: //Form Template Library FTL
23237: /////////////////////////////////
23238:
23239: FTL For Form Building Lib out of the Script, eg. 399_form_templates.txt
23240: 045 unit uPSI_VListView;                  TFormListView;
23241: 263 unit uPSI_JvProfiler32;              TProfReport
23242: 270 unit uPSI_ImgList;                  TCustomImageList
23243: 278 unit uPSI_JvImageWindow;            TJvImageWindow
23244: 317 unit uPSI_JvParserForm;             TJvHTMLParserForm
23245: 497 unit uPSI_DebugBox;                 TDebugBox
23246: 513 unit uPSI_ImageWin;                 TImageForm, TImageForm2
23247: 514 unit uPSI_CustomDrawTreeView;       TCustomDrawForm
23248: 515 unit uPSI_GraphWin;                 TGraphWinForm
23249: 516 unit uPSI_actionMain;              TActionForm
23250: 518 unit uPSI_CtlPanel;                 TAppletApplication

```

```

23251: 529 unit uPSI_MDIEdit;                                     TEditForm //RichEditApp
23252: 535 unit uPSI_CoolMain; {browser}                         TWebMainForm
23253: 538 unit uPSI_frmExportMain;                                TSynexportForm
23254: 585 unit uPSI_usniffer; { //PortScanForm}                  TSniffForm
23255: 600 unit uPSI_ThreadForm;                                 TThreadSortForm;
23256: 618 unit uPSI_delphi_arduino_Unit1;                        TLEDForm
23257: 620 unit uPSI_fpplotMain;                                  TfplotForm1
23258: 660 unit uPSI_JvDBQueryParamsForm;                         TJvQueryParamsDialog
23259: 677 unit uPSI_OptionsFrm;                                 TfrmOptions;
23260: 718 unit uPSI_MonForm;                                    TMonitorForm
23261: 742 unit uPSI_SimplePortMain;                            TPortForm1
23262: 770 unit uPSI_ovccalc;                                   TOvcCalculator //widget
23263: 810 unit uPSI_DbExcept;                                 TDbEngineErrorDlg
23264: 812 unit uPSI_GL_actorUnit1;                            TglActorForm1 //OpenGL Robot
23265: 846 unit uPSI_synhttp_daemon;                           TTCPHttpDaemon, TTCPHttpThrd, TPingThread
23266: 867 unit uPSI_Debug;                                    TfrmDebug
23267: 901 unit uPSI_JvAddPrinter;                             TJvAddPrinter //JCL
23268: 904 unit uPSI_U_HexView;                                THexForm2
23269: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)            TOscfrmMain
23270: 959 unit uPSI_AdSelCom;                                TComSelectForm
23271: 1029 unit uPSI_GPSUDemo;                               TFDemo
23272: 1031 unit uPSI_ScreenThreeeDLab;                        TFormLab3D
23273: 1051 unit uPSI_panUnit1;                                TPanForm1 //GLScene
23274: 1052 unit uPSI_DD83ul; {Arduino Tester Frm}          TDD83f1
23275: 1059 unit uPSI_uconvMain;                               TfconvMain //PS
23276: 1076 unit uPSI_CustomizeDlg;                           TCustomizeDlg / TJvAddPrinterDialog;
23277: 1111 unit uPSI_JvExceptionForm;                         TJvErrorDialog //ShowException
23278: 1113 unit uPSI_MTMainForm;                            TvtMainForm
23279: 1119 unit uPSI_CPortTrmSet;                           TComTrmSetForm
23280:
23281:
23282: FormTemplates with <Ctrl> J
23283:
23284: [myformtemplate | formtemplate statement | Borland.EditOptions.Pascal] //with
23285: [myFastForm | class declaration | Borland.EditOptions.Pascal] //Dialog Form
23286: [myForm | class declaration | Borland.EditOptions.Pascal] //with Events
23287: [aForm | class declaration | Borland.EditOptions.Pascal] //single Form
23288:
23289:
23290: procedure SIRegister_JvDesignUtils(CL: TPSPascalCompiler);
23291: begin
23292:   Function DesignClientToParent( const APt : TPoint; AControl, AParent : TControl ) : TPoint';
23293:   Function DesignMin( AA, AB : Integer ); Function DesignMax( AA, AB: Integer): Integer';
23294:   Function DesignRectWidth( const ARect : TRect ) : Integer';
23295:   Function DesignRectHeight( const ARect : TRect ) : Integer';
23296:   Function DesignValidateRect( const ARect : TRect ) : TRect';
23297:   Function DesignNameIsUnique( AOwner : TComponent; const AName : string ) : Boolean';
23298:   Function DesignUniqueName( AOwner : TComponent; const AClassName : string ) : string';
23299:   Procedure DesignPaintRubberbandRect( AContainer : TWinControl; ARect : TRect; APenStyle : TPenStyle );
23300:   Procedure DesignPaintGrid(ACanvas:TCanvas;const ARect:TRect;ABackClr:TColor;AGridClr:TColor;ADivPixels: Integer)
23301:   Procedure DesignPaintRules( ACanvas : TCanvas; const ARect:TRect; ADivPixels:Integer; ASubDivs: Boolean );
23302:   Procedure DesignSaveComponentToStream( AComp : TComponent; AStream : TStream );
23303:   Function DesignLoadComponentFromStream(AComp:TComponent;AStream:TStream;
23304:     AOnError:TReaderError ):TComponent;
23304:   Procedure DesignSaveComponentToFile( AComp : TComponent; const AFileName : string );
23305:   Procedure DesignLoadComponentFromFile(AComp:TComponent; const AFileName: string; AOnError:TReaderError );
23306: end;
23307:
23308: ex.:with TEditForm.create(self) do begin
23309:   caption:= 'Template Form Tester';
23310:   FormStyle:= fsStayOnTop;
23311:   with editor do begin
23312:     Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mx2.rtf'
23313:     SelStart:= 0; Modified:= False;
23314:   end;
23315: end;
23316: with TWebMainForm.create(self) do begin
23317:   URLs.Text:= 'http://www.kleiner.ch';
23318:   URLsClick(self); Show;
23319: end;
23320: with TSynexportForm.create(self) do begin
23321:   Caption:= 'Synexport HTML RTF tester';
23322:   Show;
23323: end;
23324: with TThreadSortForm.create(self) do begin
23325:   showmodal; free;
23326: end;
23327: with TCustomDrawForm.create(self) do begin
23328:   width:=820; height:=820;
23329:   image1.height:= 600; //add properties
23330:   image1.picture.bitmap:= image2.picture.bitmap;
23331:   //SelectionBackground1Click(self) CustomDraw1Click(self);
23332:   Background1.click;
23333:   bitmap1.click; Tile1.click;
23334:   Showmodal;
23335:   Free;
23336: end;
23337: with TfplotForm1.Create(self) do begin

```

```

23338:     BtnPlotClick(self);
23339:     Showmodal; Free;
23340:   end;
23341:   with TOvcCalculator.create(self) do begin
23342:     parent:= aForm; //free;
23343:     setbounds(550,510,200,150);
23344:     displaystr:= 'maXcalc';
23345:   end;
23346:   with THexForm2.Create(self) do begin
23347:     ShowModal;
23348:     Free;
23349:   end;
23350:
23351: function CheckBox: string;
23352: var idHTTP: TIDHTTP;
23353: begin
23354:   result:= 'version not found';
23355:   if IsInternet then begin
23356:     idHTTP:= TIdHTTP.Create(NIL);
23357:     try
23358:       result:= idHTTP.Get(MXVERSIONFILE2);
23359:       result:= result[1]+result[2]+result[3]+result[4]+result[5];
23360:       if result = MBVER2 then begin
23361:         //Speak(' A new Version '+vstr+' of max box is available! ');
23362:         result:= ('!!! OK. You have latest Version: '+result+ ' available at '+MXSITE);
23363:       end; //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
23364:     finally
23365:       idHTTP.Free
23366:     end;
23367:   end;
23368: end;
23369:
23370: //Runtimer Spec Functions Edition 190
23371: function ApWinExecAndWait32(FileNamed:PChar; CommandLine:PChar; Visibility:Integer):Integer;
23372: function KillTask(ExeFileName: string): Integer;
23373: procedure KillProcess(hWindowHandle: HWND);
23374: function FindWindowByTitle(WindowTitle: string): Hwnd;
23375: function IntToFloat(i: Integer): double;
23376: function AddThousandsSeparator(S: string; myChr: Char): string;
23377: function mciSendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
23378: procedure FormAnimation(Sender: TObject; adelay: integer);
23379: procedure LoadResourceFile2(aFile:string; ms:TMemoryStream);
23380: function putBinResTo(binresname: pchar; newpath: string): boolean;
23381: procedure ExecuteHyperlink(Sender:TOObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:string);
23382: function IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:string; MouseX,MouseY:Integer;var HyperLink:string):Bool
23383: Function GetWindowThreadProcessId( hWnd : HWND; var dwProcessId : DWORD ) : DWORD; '
23384: Function GetWindowTask( hWnd : HWND ) : THandle;';
23385: Function LoadBitmap( hInstance : HINST; lpBitmapName : PChar ) : HBITMAP;';
23386: Function GetCommConfig(hCommDev: THandle; var lpCC: TCommConfig; var lpdwSize: DWORD): BOOL;';
23387: function WinExecAndWait32(FileNamed: string; Visibility: Integer): Longword;
23388: Function MakeHash( const s : TbtString ) : Longint;';
23389: Function GetUsedUnitList( list : TStringlist ): string)';
23390: function ConsoleCapture(const _dirName, _exeName, _cmdLine: string; amemo: TStringlist): Boolean;';
23391: function ConsoleCaptureDOS(const _dirName, _exeName, _cmdLine: string; amemo: TStrings): Boolean;';
23392: srlist:= TStringlist.create;
23393:   ConsoleCapture('C:\', 'cmd.exe', '/c dir .*.',srlist);
23394:   writeln(srlist.text); srlist.Free;
23395: function RunCaptured(const _dirName, _exeName, _cmdLine: string; amemo: TStringlist): Boolean;';
23396: Function SamePropTypeName( const Name1, Name2 : ShortString ) : Boolean;';
23397: Function FloatToStrEx( Value : Extended ) : string)';
23398: Function StrToFloatEx( const S : string ) : Extended)';
23399: Procedure PerformanceDelayMS(ams: integer); //microsecond resolution delay!
23400: //http://www.swissdelphicenter.ch/en/showcode.php?id=2179
23401: function ExecuteProcess(FileName: string;Visibility:Integer; BitMask:Integer; Synch:Boolean):Longword;
23402: Function
23403:   ExecuteMultiProcessor(FileName:string;Visibility:Integer;BitMask:Integer;Synch:Boolean):Longword;';
23404:   if ExecuteMultiProcessor('notepad.exe', SW_SHOW, 2, true) = 0 then
23405:     writeln('Multiprocessing Runs on CPU 2');
23406: procedure StartServiceAfterInstall(aserv: TSERVICE);
23407: Function GetDllVersion2(DllName: string; var DLLVersionInfo: TDLLVersionInfo): Boolean;
23408: procedure SendCopyMessage(amess, astation: string); //communicate process-spanned with WM_COPYDATA
23409: function BrowseComputer2(DialogTitle: string; var CompName: string; bNewStyle: Boolean): Boolean;
23410: function ChangeAlphaTo(input: string; aoffset: byte): string)';
23411: function CheckIBAN(iban: string): Boolean;';
23412: Function CreateIDStack; //instance to IdStack of CreateStack
23413: Function GetRecordCount(DataSet: TBDEDataSet): Longint;';
23414: Function CountPos(const subtxt: string; Text: string): Integer;
23415: procedure HTMLtoRTF(html: string; var rtf: TRichedit);
23416: Function ADOConnectionString(ParentHandle:THandle; InitialString:WideString;out NewString: string):Boolean;
23417: procedure ShowEOLEException(AExc: EOleException; Query: String)';
23418: function UpdateBlob(Connection: TADOConnection; Spalte: String; Tabelle: String; Where: String; var ms: TMemoryStream): Boolean; ..save HTML pages as MHTML (HTML Archiv Format)
23419:   http://www.swissdelphicenter.ch/en/showcode.php?id=2300
23420: function SaveToMHT(const AUrl, AFileName: string; AShowErrorMessage: Boolean = False): Boolean;
23421: function FileType2MimeType(const AFileName: string): string;
23422: function DownloadURL_NOCache(const aUrl: string; var s: String): Boolean;
23423: Function IsCOMObjectActive(ClassName: string): Boolean;

```

```

23424: Procedure CopyHTMLToClipBoard(const str: string; const htmlStr: string = '');
23425: function CheckCreditCard(c: string): Integer;
23426:   0: Card is invalid or unknown 1: is a AmEx 2: is a Visa 3: is a valid MasterCard
23427:
23428: function getFormRes(classname: string): string; //shows DFM Res of Exe!
23429: procedure OutputDebugString(PChar(Format('[%s][%s] %s',[aCaption, GetFormatDT(StartDT), aText])));
23430: procedure ScanNetworkResources(ResourceType, DisplayType: DWord; List: TStrings);
23431: //ScanNetworkResources(RESOURCETYPE_DISK, RESOURCEDISPLAYTYPE_SERVER, ListBox1.Items);
23432: function ComputePEChecksum(FileName: string): DWORD;
23433: if not DynamicDllCallName(user32, 'LockWorkStation', true, returned, parameters) then begin
23434: function DynamicDllCallNames(Dll: String; const Name: String; HasResult: Boolean; var Returned: Cardinal;
23435: const Parameters: array of string): Boolean; );
23436: Procedure GetMIDASAppServerList(List: TStringList; const RegCheck : string)');
23437: procedure SQLDropField(dbName, tblName, fldName: String); {Field Name to Drop}
23438: type TCastType = (ctSmallInt, ctInteger, ctDecimal, ctNumeric, ctFloat,
23439:                      ctChar, ctVarChar, ctDate, ctBoolean, ctBLOB, ctTime,
23439:                      ctTimeStamp, ctMoney, ctAutoInc, ctBytes);
23440: {Blob definition type 1 = Memo, 2 = Binary, 3 = Formatted Memo, 4 = OLE Object, 5 = Graphic}
23441:
23442: procedure SQLAddField(dbName,tblName,fldName: String; fldType: TCastType; fldLength, precisOrBlobLen,
23443: scaleOrBlobType: Integer);
23443: const
23444: UrlGeoLookupInfo ='http://ipinfodb.com/ip_query.php?timezone=true&ip=%s';
23445: UrlGeoLookupInfo2 ='http://backup.ipinfodb.com/ip_query.php?timezone=true&ip=%s'; //backup
23446:
23447: procedure GetGeoInfo(const IpAddress : string;var GeoInfo : TGeoInfo; const UrlGeoLookupInfo: string);
23448: CL.AddTypeS('TGeoInfo', 'record status: string; countrycode : '
23449:   +string; countrynam : string; regioncode : string; city : string; zippostalcode : string; latitude '
23450:   +: string; longitude : string; timezonename : string; gmtoffset: string; isdst: string; end');
23451:
23452: procedure SIRegister_ubigFloatV3(CL: TPSPascalCompiler);
23453: begin
23454:   CL.AddTypeS('TMaxSig', 'integer');
23455:   CL.AddTypeS('TView', '( normal, Scientific )');
23456:   SIRegister_TFloatInt(CL); SIRegister_TBigFloat(CL);
23457: end;
23458:
23459: procedure SIRegister_UBigIntsV4(CL: TPSPascalCompiler);
23460: begin TDigits', 'array of int64'); SIRegister_TInteger(CL);
23461: Procedure SetBaseVal( const newbase : integer)';
23462: Function GetBasePower : integer)';
23463: Function GetBase : integer)';
23464: Procedure SetThreadSafe( newval : boolean)';
23465: end;
23466:
23467: function BigDiv(aone, atwo: string): string;
23468: var tbig1, tbig2: TInteger;
23469: begin
23470:   tbig1:= TInteger.create(10);
23471:   tbig2:= TInteger.create(10);
23472:   try
23473:     tbig1.assign2(atwo)
23474:     tbig2.assign2(aone)
23475:     tbig2.Divide(tbig1)
23476:   finally
23477:     result:= tbig2.Convert.ToDecimalString(false)
23478:     tbig1.Free;
23479:     tbig2.free;
23480:   end;
23481: end;
23482:
23483: procedure SIRegister_UDict(CL: TPSPascalCompiler);
23484: begin CL.AddConstantN('dichighletter','String').SetString( 'z' );
23485:   SIRegister_TDicForm(CL);
23486:   SIRegister_TDic(CL);
23487: end;
23488:
23489: procedure SetArrayLength2Char2(var arr: T2CharArray; asizel, asize2: integer);
23490: var i: integer;
23491: begin setlength(arr, asizel);
23492:   for i:= 0 to asizel do SetLength(arr[i], asize2);
23493: end;
23494:
23495: procedure SIRegister_UP10Build(CL: TPSPascalCompiler);
23496: begin Procedure ParseFunction(FunctionString:string; Variables:TStringlist; FunctionOne,FunctionTwo:
23497: TStringList; UsePascalNumbers : boolean; var FirstTOP: {TOBJECT}POperation; var Error : boolean)');
23498: end;
23499: procedure SIRegister_ComObj2(CL: TPSPascalCompiler);
23500: begin
23501:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TComObjectFactory');
23502:   SIRegister_TComServerObject(CL);
23503:   SIRegister_ADOConst(CL);
23504:   FieldTypeNames: array[0..41] of string = (
23505:     'Unknown', 'String', 'SmallInt', 'Integer', 'Boolean', 'Float',
23506:     'Currency', 'BCD', 'Date', 'Time', 'DateTime', 'Bytes', 'VarBytes',
23507:     'AutoInc', 'Blob', 'Memo', 'Graphic', 'FmtMemo', 'ParadoxOle',
23508:     'dBaseOle', 'TypedBinary', 'Cursor', 'FixedChar', 'WideString',
23509:     'LargeInt', 'ADT', 'Array', 'Reference', 'DataSet', 'HugeBlob', 'HugeClob',

```

```

23510:   'Variant', 'Interface', 'Dispatch', 'Guid', 'SQLTimeStamp', 'FMTBcdField',
23511:   'FixedWideChar', 'WideMemo', 'SQLTimeStamp', 'String');
23512: CL.AddTypeS('TFactoryProc', 'Procedure (Factory : TComObjectFactory)');
23513: CL.AddTypeS('TCallingConvention','(ccRegister, ccDecl, ccPascal, ccStdCall, ccSafeCall)');
23514: SIRegister_TComClassManager(CL);
23515: SIRegister_IServerExceptionHandler(CL); SIRegister_TComObject(CL);
23516: //CL.AddTypeS('TComClass', 'class of TComObject');
23517: CL.AddTypeS('TClassInstancing', '(ciInternal,ciSingleInstance,ciMultiInstance )');
23518: CL.AddTypeS('TThreadingModel', '(tmSingle, tmApartment, tmFree, tmBoth, tmNeutral )');
23519: SIRegister_TComObjectFactory(CL); SIRegister_TTypedComObject(CL);
23520: //CL.AddTypeS('TTypedComClass', 'class of TTtypedComObject');
23521: SIRegister_TTypedComObjectFactory(CL);
23522: CL.AddTypeS('TConnectEvent2', 'Procedure(const Sink : IUnknown; Connecting : Boolean)');
23523: CL.AddClassN(CL.FindClass('TOBJECT'), 'TAutoObjectFactory');
23524: SIRegister_TAutoObject(CL); //TAutoObject2 ?? in OleAuto and ComObj
23525: //CL.AddTypeS('TAutoClass', 'class of TAtoObject');
23526: SIRegister_TAutoObjectFactory(CL); SIRegister_TAutoIntfObject(CL);
23527: //CL.AddClassN(CL.FindClass('TOBJECT'), 'EOleError');
23528: CL.AddClassN(CL.FindClass('Exception'), 'EOleError');
23529: SIRegister_EOLError(CL); SIRegister_EOLEException(CL);
23530: CL.AddClassN(CL.FindClass('TOBJECT'), 'EOleRegistrationError');
23531: //Procedure DispatchInvoke( const Dispatch : IDispatch; CallDesc : PCallDesc; DispIDs : PDispIDList;
Params : Pointer; Result : PVariant);
23532: //Procedure DispatchInvokeError( Status : Integer; const ExcepInfo : TExcepInfo);
23533: //Function HandleSafeCallException( ExceptObject : TObject; ExceptAddr : Pointer; const ErrorIID : TGUID;
const ProgID, HelpFileName : WideString) : HResult';
23534: Function CreateComObject( const ClassID : TGUID) : IUnknown';
23535: Function CreateRemoteComObject( const MachineName : WideString; const ClassID : TGUID) : IUnknown';
23536: //Function CreateOleObject(const ClassName : string): IDispatch';
23537: //Function GetActiveOleObject(const ClassName:string): IDispatch';
23538: Procedure OleError2( ErrorCode : HResult)';
23539: Procedure OleCheck2( Result : HResult)';
23540: Function StringToGUID2( const S : string) : TGUID';
23541: Function GUIDToString2( const ClassID : TGUID) : string';
23542: Function ProgIDToClassID2( const ProgID : string) : TGUID';
23543: Function ClassIDToProgID2( const ClassID : TGUID) : string';
23544: Procedure CreateRegKey(const Key,ValueName,Value:string;RootKey:DWord)';
23545: Procedure DeleteRegKey(const Key : string; RootKey : DWord)';
23546: Function GetRegStringValue( const Key, ValueName : string; RootKey : DWord) : string';
23547: //Function StringToLPOLESTR( const Source : string) : PoleStr';
23548: Procedure RegisterComServer( const DLLName : string)';
23549: Procedure RegisterAsService( const ClassID, ServiceName : string)';
23550: Function CreateClassID2 : string)';
23551: Procedure InterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var Connection:Longint)';
23552: Procedure InterfaceDisconnect( const Source : IUnknown; const IID : TIID; var Connection : Longint)';
23553: Function GetDispatchPropValue(Disp: IDispatch; DispID : Integer) : OleVariant';
23554: Function GetDispatchPropValue1(Disp: IDispatch; Name : WideString) : OleVariant';
23555: Procedure SetDispatchPropValue2(Disp: IDispatch; DispID : Integer; const Value : OleVariant)';
23556: Procedure SetDispatchPropValue3(Disp: IDispatch; Name : WideString; const Value : OleVariant)';
23557: Function ComClassManager : TComClassManager)';
23558: // from ADODB OLE Utils
23559: CL.AddTypeS('TOleEnum', 'LongWord'); //DataTypeEnum = TOleEnum;
23560: CL.AddTypeS('DataTypeEnum', 'TOleEnum');
23561: Function CreateADOOObject( const ClassID : TGUID) : IUnknown';
23562: Function ADOTypeToFieldType( const ADOType: DataTypeEnum; EnableBCD : Boolean) : TFieldType';
23563: Function FieldTypeToADOType( const FieldType: TFieldType):DataTypeEnum';
23564: Function StringToVarArray( const Value : string) : OleVariant';
23565: Function VarDataSize( const Value : OleVariant) : Integer';
23566: Function OleEnumToOrd( OleEnumArray : array of TOleEnum; Value : TOleEnum) : Integer';
23567: Function GetStates( State : Integer) : TObjectStates';
23568: Function ExecuteOptionsToOrd(ExecuteOptions:TExecuteOptions) : Integer';
23569: Function OrdToExecuteOptions(Options: Integer) : TExecuteOptions';
23570: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString';
23571: Function GetFilterStr( Field : TField; Value : Variant, Partial : Boolean) : WideString';
23572: Function FieldListChecksum( DataSet : TDataSet) : Integer';
23573: function GlobalAllocString(s: AnsiString): HGlobal;
23574: function ScanTime(const S: string; var Pos: Integer; var Time: TDateTime): Boolean;
23575: function ScanChar(const S: string; var Pos: Integer; Ch: Char): Boolean;
23576: function ScanNumber(const S: string; var Pos: Integer; var Number: Word): Boolean;
23577: function ScanString(const S: string; var Pos: Integer; const Symbol: string): Boolean;
23578: procedure LV_InsertFiles(strPath: string; ListView: TListView; ImageList: TImageList);
23579: Function GetPasteLinkInfo( var Service : string; var Topic : string; var Item : string) : Boolean';
23580: function IPTOHostName(const IP: string): string;
23581: procedure GetZoneIcon(IconPath: string; var Icon: TIcon);
23582: function GetZoneAttributes(const URL: string): TZoneAttributes;
23583: //unit uPSI_PsAPI;
23584: procedure CGITester; //CGI will take name and email address from command line and place it into HTML
23585: procedure CreateBrowserOnForm(aform: TForm; aurl: string);
23586: procedure WebOnForm(aform: TForm; aurl: string);
23587: procedure WebToForm(aform: TForm; aurl: string);
23588: procedure SearchAndHighlightWebText(aform: TForm; aurl: string; aText: string);
23589: procedure SaveImagesOnWeb(aurl, apath: string);
23590: function GetProcessNameFromWnd(Wnd: HWND): string; //get EXE path from window handle
23591: function getallEvents(aform: TForm): TStringlist;
23592: procedure GetKLList(List: TStrings);
23593: procedure GetKeyboardList(List: TStrings);
23594: function SetSuspendState(Hibernate: Boolean;ForceCritical: Boolean;DisableWakeEvent: Boolean): boolean;
23595:           call: SetSuspendState(True, False, False);

```

```

23596: function ServiceRunning(sMachine, sService: PChar): Boolean;
23597: function isServiceRunning(sMachine, sService: PChar): Boolean;
23598: Procedure CloseOpenSockets( Sockets : array of TIdStackSocketHandle); //SIRegister_SvrHTTPIndy - Linux
23599: Procedure TransformError( const Msg : string');
23600: Procedure StringToFile2( const S, FileName : string');
23601: Function GetXMLData( DataSet : TClientDataSet) : string';
23602: Procedure EditComTerminal( ComTerminal : TCustomComTerminal); //TComTrmSetForm
23603: function search_adapter_key_networkcard: string; //at registry
23604: function getNetworkCard: string';
23605: function GetMacAddresses2(const Machine: string; const Addresses: TStrings): Integer;
23606: function ConnectDrive(_drvLetter: string; _netPath: string; _showError:Boolean; _reconnect:Boolean):DWORD;
23607: function
  ConnectPrinterDevice(_lpPort:string;_netPath:string;_showError:Boolean;_reconnect:Boolean):DWORD;
23608: function DisconnectNetDrive(_locDrive: string; _showError: Boolean; _force: Boolean; _save:Boolean): DWORD;
23609:   //ConnectDrive('k:', '\Servername\C', True, True);
23610:   //DisconnectNetDrive('k:', True, True, True);
23611: function GetConnectionKind(var strKind: string): Boolean; MODEM = 1; LAN = 2; PROXY = 4; BUSY = 8;
23612: function DownloadJPGToBitmap(const URL : string; ABitmap: TBitmap): Boolean;
23613: procedure GetImageLinks(AURL: string; AList: TStrings);
23614:
23615:
23616: IPToHostName
23617: //from ADOInt.pas // _ToleEnum = type LongWord;
23618: CL.AddTypeS('CursorOptionEnum', '_ToleEnum');
23619: CL.AddInterface(CL.FindInterface('IUNKNOWN'),_Recordset, '_Recordset');
23620: //CL.AddTypeS('_RecordsetDisp', 'dispinterface');
23621: CL.AddInterface(CL.FindInterface('IUNKNOWN'),_Command, '_Command');
23622: CL.AddInterface(CL.FindInterface('IUNKNOWN'),_Connection, '_Connection');
23623: // SIRegister_Recordset(CL); // SIRegister_Command(CL);
23624: Const IID_Recordset20', 'TGUID').SetString('{0000054F-0000-0010-8000-00AA006D2EA4}''']);
23625: CL.AddConstantN('IID_Recordset','string').SetString( '00000555-0000-0010-8000-00AA006D2EA4');
23626: //test with stringToGUID
23627: //CL.AddConstantN('CLASS_Command','TGUID').SetString( '{00000507-0000-0010-8000-00AA006D2EA4}');
23628: // CL.AddConstantN('CLASS_Recordset','TGUID').SetString( '{00000535-0000-0010-8000-00AA006D2EA4}');
23629: { CL.AddTypeS('Connection', '_Connection');
23630: CL.AddTypeS('Command', '_Command'); CL.AddTypeS('Recordset', '_Recordset');
23631: CL.AddTypeS('Parameter', '_Parameter'); CL.AddTypeS('DataSpace', 'IDataspace');
23632: CL.AddTypeS('SearchDirection', 'SearchDirectionEnum');
23633: //CL.AddTypeS('Command', '_Command'); //CL.AddTypeS('Recordset', '_Recordset');
23634: end;
23635:
23636: procedure SIRegister_DIUtils(CL: TPSPPascalCompiler);
23637: begin
23638: //CL.AddConstantN('CRLF','String').SetString( #$0D#$0A);
23639: CL.AddConstantN('REPLACEMENT_CHARACTER','LongWord').SetUInt( $FFFD);
23640: CL.AddConstantN('HANGUL_SBase','LongWord').SetUInt( $AC00);
23641: CL.AddConstantN('HANGUL_LBase','LongWord').SetUInt( $1100);
23642: CL.AddConstantN('HANGUL_VBase','LongWord').SetUInt( $1161);
23643: CL.AddConstantN('HANGUL_TBase','LongWord').SetUInt( $11A7);
23644: CL.AddConstantN('HANGUL_LCount','LongInt').SetInt( 19);
23645: CL.AddConstantN('HANGUL_VCount','LongInt').SetInt( 21);
23646: CL.AddConstantN('HANGUL_TCount','LongInt').SetInt( 28);
23647: CL.AddConstantN('KEY_WOW64_32KEY','LongWord').SetUInt( $0200);
23648: CL.AddConstantN('KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
23649: CL.AddConstantN('KEY_WOW64_RES','LongWord').SetUInt( $0300);
23650: CL.AddTypeS('TAnsiCharSet', 'set of AnsiChar');
23651: CL.AddTypeS('TIsoDate', 'Cardinal');
23652: CL.AddTypeS('TJulianDate', 'Integer');
23653: //CL.AddTypeS('PJulianDate', '^TJulianDate // will not work');
23654: CL.AddTypeS('TValidateCharFunc', 'function(const c: Char): Boolean;');
23655: // TValidateCharFunc = function( const c: Char): Boolean;
23656: CL.AddTypeS('TPProcedureEvent', 'Procedure');
23657: CL.AddConstantN('MT19937_N','LongInt').SetInt( 624);
23658: CL.AddConstantN('MT19937_M','LongInt').SetInt( 397);
23659: SIRegister_MT19937(CL);
23660: //SIRegister_TWideStrBuf(CL);
23661: CL.AddTypeS('TDITextLineBreakStyle', '( tlbsLF, tlbsCRLF, tlbsCR )');
23662: Function AdjustLineBreaksW( const s : UnicodeString; const Style : TDITextLineBreakStyle) : UnicodeString';
23663: Function BrightenColor( const Color : Integer; const amount : Byte) : Integer';
23664: Function BSwap4( const Value : Cardinal) : Cardinal';
23665: Function BSwap5( const Value : Integer) : Integer';
23666: //Function BufCompNumIW( p1 : PWideChar; l1 : Integer; p2 : PWideChar; l2 : Integer) : Integer';
23667: Function BufSameA( p1, p2 : PChar; l : Cardinal) : Boolean';
23668: // Function BufSameW( p1, p2 : PWideChar; l : Cardinal) : Boolean';
23669: Function BufSameIA( p1, p2 : PChar; l : Cardinal) : Boolean';
23670: //Function BufSameIW( p1, p2 : PWideChar; l : Cardinal) : Boolean';
23671: Function BufPosCharA( const Buf : PChar; l : Cardinal; const c : AnsiChar; const Start : Cardinal) : Integer';
23672: Function BufPosCharsA(const Buf:PChar; l:Cardinal;const Search:TAnsiCharSet;const Start: Cardinal) : Integer';
23673: Function BufStrSame( const Buf : PChar; const BufCharCount : Cardinal; const s : string) : Boolean';
23674: Function BufStrSameA( const Buf : PChar; const BufCharCount : Cardinal; const s : RawByteString) : Boolean';
23675: Function BufStrSameI( const Buf : PChar; const BufCharCount : Cardinal; const s : string) : Boolean';
23676: Function BufStrSameIA( const Buf : PChar; const BufCharCount : Cardinal; const s : RawByteString) : Boolean';
23677: Function diChangeFileExt(const FileName,Extension: string): string';
23678: Function ChangeFileExtA( const FileName, Extension : AnsiString) : AnsiString';

```

```

23679: Function ChangeFileExtW( const FileName, Extension : UnicodeString ) : UnicodeString');
23680: //Function CharDecomposeCanonicalW( const c : WideChar ) : PCharDecompositionW';
23681: Function CharDecomposeCanonicalStrW( const c : WideChar ) : UnicodeString');
23682: //Function CharDecomposeCompatibleW( const c : WideChar ) : PCharDecompositionW';
23683: Function CharDecomposeCompatibleStrW( const c : WideChar ) : UnicodeString');
23684: Function CharIn8( const c, t1, t2 : WideChar ) : Boolean');
23685: Function CharIn9( const c, t1, t2, t3 : WideChar ) : Boolean');
23686: Procedure ConCatBuf( const Buffer : PChar; const CharCount : Cardinal; var d : string; var InUse : Cardinal);
23687: Procedure ConCatBufA(const Buffer:PChar;const AnsiCharCount:Cardinal;var d:RawByteString;var InUse:Cardinal);
23688: Procedure ConCatChar( const c : Char; var d : string; var InUse : Cardinal)');
23689: Procedure ConCatCharA( const c : AnsiChar; var d : RawByteString; var InUse : Cardinal)');
23690: // Procedure ConCatCharW( const c : WideChar; var d : UnicodeString; var InUse : Cardinal)');
23691: Procedure ConCatStr( const s : string; var d : string; var InUse : Cardinal)');
23692: Procedure ConCatStrA( const s : RawByteString; var d : RawByteString; var InUse : Cardinal)');
23693: //Procedure ConCatStrW( const w : UnicodeString; var d : UnicodeString; var InUse : Cardinal)');
23694: Function diCountBitsSet( const Value : Integer ) : Byte');
23695: //Function Crc32OfStrA( const s : RawByteString ) : TCrc32');
23696: //Function Crc32OfStrW( const s : UnicodeString ) : TCrc32');
23697: Function CurrentDay : Word');
23698: Function CurrentJulianDate : TJ JulianDate');
23699: Function CurrentMonth : Word');
23700: Function CurrentQuarter : Word');
23701: Function diCurrentYear : Integer');
23702: Function DarkenColor( const Color : Integer; const amount : Byte ) : Integer');
23703: Function diDeletefile( const FileName : string ) : Boolean');
23704: Function DeleteFileA( const FileName : AnsiString ) : Boolean');
23705: //Function DeleteFileW( const FileName : UnicodeString ) : Boolean');
23706: Function diDirectoryExists( const Dir : string ) : Boolean');
23707: Function DirectoryExistsA( const Dir : AnsiString ) : Boolean');
23708: //Function DirectoryExistsW( const Dir : UnicodeString ) : Boolean');
23709: Function diDiskFree( const Dir : string ) : Int64');
23710: Function DiskFreeA( const Dir : AnsiString ) : Int64');
23711: //Function DiskFreeW( const Dir : UnicodeString ) : Int64');
23712: Function diExpandFileName( const FileName : string ) : string');
23713: Function ExpandFileNameA(const FileName: AnsiString): AnsiString');
23714: //Function ExpandFileNameW( const FileName : UnicodeString ) : UnicodeString');
23715: Procedure diExcludeTrailingPathDelimiterA( var s : string)');
23716: Procedure ExcludeTrailingPathDelimiterA( var s : RawByteString)');
23717: //Procedure ExcludeTrailingPathDelimiterW( var s : UnicodeString)');
23718: Function diExtractFileDrive( const FileName : string ) : string');
23719: Function ExtractFileDriveA( const FileName : RawByteString ) : RawByteString');
23720: //Function ExtractFileDriveW( const FileName : UnicodeString ) : UnicodeString');
23721: Function diExtractFileExt( const FileName : string ) : string');
23722: Function ExtractFileExtA( const FileName : RawByteString ) : RawByteString');
23723: //Function ExtractFileExtW( const FileName : UnicodeString ) : UnicodeString');
23724: Function diExtractFileName( const FileName : string ) : string');
23725: Function ExtractFileNameA(const FileName: AnsiString): AnsiString');
23726: //Function ExtractFileNameW( const FileName : UnicodeString ) : UnicodeString');
23727: Function diExtractFilePath( const FileName : string ) : string');
23728: Function ExtractFilePathA( const FileName : RawByteString ) : RawByteString');
23729: //Function ExtractFilePathW( const FileName : UnicodeString ) : UnicodeString');
23730: Function ExtractNextWord10(const s: string; const ADelimiter : Char; var AStartIndex : Integer) : string');
23731: Function ExtractNextWordAll(const s: RawByteString; const ADelimiter : AnsiChar; var AStartIndex : Integer) : RawByteString');
23732: Function ExtractNextWordW12(const s : UnicodeString; const ADelimiter : WideChar; var AStartIndex : Integer) : UnicodeString');
23733: Function ExtractNextWord13(const s:string;const ADelimiters:TAnsiCharSet;var AstartIndex:Integer):string');
23734: Function ExtractNextWordA14(const s:RawByteString; const ADelimiters:TAnsiCharSet; var AstartIndex : Integer) : RawByteString');
23735: Function diExtractWord( const Number : Cardinal; const s : RawByteString; const Delimiters : TAnsiCharSet ) : RawByteString');
23736: Function ExtractWordA( const Number : Cardinal; const s : RawByteString; const Delimiters : TAnsiCharSet ) : RawByteString');
23737: Function ExtractWordStartsA( const s : RawByteString; const MaxCharCount : Cardinal; const WordSeparators : TAnsiCharSet ) : RawByteString');
23738: Function diFileExists( const FileName : string ) : Boolean');
23739: Function FileExistsA( const FileName : AnsiString ) : Boolean');
23740: //Function FileExistsW( const FileName : UnicodeString ) : Boolean');
23741: Function diGCD( x, y : Cardinal ) : Cardinal');
23742: Function diGetTempFolder : string');
23743: Function GetTempFolderA : AnsiString');
23744: //Function GetTempFolderW : UnicodeString');
23745: Function diGetUserName( out UserName : string ) : Boolean');
23746: Function GetUserNameA( out UserName : AnsiString ) : Boolean');
23747: //Function GetUserNameW( out UserName : UnicodeString ) : Boolean');
23748: Function HexCodePointToInt( const c : Cardinal ) : Integer');
23749: Function diHexToInt( const s : string ) : Integer');
23750: Function HexToIntA( const s : RawByteString ) : Integer');
23751: //Function HexToIntW( const s : UnicodeString ) : Integer');
23752: Function BufHexToInt( p : PChar; l : Cardinal ) : Integer');
23753: Function BufHexToIntA( p : PChar; l : Cardinal ) : Integer');
23754: //Function BufHexToIntW( p : PWideChar; l : Cardinal ) : Integer');
23755: Procedure IncludeTrailingPathDelimiterByRef( var s : string)');
23756: Procedure IncludeTrailingPathDelimiterByRefA(var s:RawByteString)');
23757: //Procedure IncludeTrailingPathDelimiterByRefW(var w:UnicodeString)');

```

```

23758: Function IntToHex16( const Value : Integer; const Digits : NativeInt ) : string');
23759: Function IntToHex17( const Value : Int64; const Digits : NativeInt ) : string');
23760: Function IntToHex18( const Value : UInt64; const Digits : NativeInt ) : string');
23761: Function IntToHexA( Value : UInt64; const Digits : NativeInt ) : RawByteString');
23762: //Function IntToHexW( Value : UInt64; const Digits : NativeInt ) : UnicodeString');
23763: Function IntToStrA19( const i : Integer ) : RawByteString');
23764: Function IntToStrW20( const i : Integer ) : UnicodeString');
23765: Function IntToStrA21( const i : Int64 ) : RawByteString');
23766: Function IntToStrW22( const i : Int64 ) : UnicodeString');
23767: Function CharDecomposeHangulW( const c : WideChar ) : UnicodeString');
23768: Function diIsPathDelimiter( const s : string; const Index : Cardinal ) : Boolean');
23769: Function IsPathDelimiterA( const s : RawByteString; const Index : Cardinal ) : Boolean');
23770: //Function IsPathDelimiterW( const s : UnicodeString; const Index : Cardinal ) : Boolean');
23771: Function IsPointInRect( const Point : TPoint; const Rect : TRect ) : Boolean');
23772: Function JulianDateToIsoDateStr( const Julian : TJ JulianDate ) : string');
23773: Function JulianDateToIsoDateStrA( const Julian : TJ JulianDate ) : RawByteString');
23774: //Function JulianDateToIsoDateStrW( const Julian : TJ JulianDate ) : UnicodeString');
23775: Function LeftMostbit( Value : Cardinal ) : ShortInt');
23776: Function LeftMostBit2( Value : UInt64 ) : ShortInt');
23777: //Function MakeMethod( const AData, ACode : Pointer ) : TMethod');
23778: Function StrIsEmpty( const s : string ) : Boolean');
23779: Function StrIsEmptyA( const s : RawByteString ) : Boolean');
23780: //Function StrIsEmptyW( const s : UnicodeString ) : Boolean');
23781: Function PadLeftA( const Source : RawByteString; const Count : Cardinal; const c : AnsiChar ) : RawByteString');
23782: //Function PadLeftW( const Source : UnicodeString; const Count : Cardinal; const c : WideChar ) : UnicodeString');
23783: Function PadRightA( const Source : RawByteString; const Count : Cardinal; const c : AnsiChar ) : RawByteString');
23784: //Function PadRightW( const Source : UnicodeString; const Count : Cardinal; const c : WideChar ) : UnicodeString');
23785: Function ProperCase( const s : string ) : string');
23786: Function ProperCaseA( const s : RawByteString ) : RawByteString');
23787: //Function ProperCaseW( const s : UnicodeString ) : UnicodeString');
23788: Procedure ProperCaseByRefA( var s : RawByteString );
23789: //Procedure ProperCaseByRefW( var s : UnicodeString );
23790: Function RegReadRegisteredOrganization( const Access : REGSAM ) : string');
23791: Function RegReadRegisteredOrganizationA( const Access : REGSAM ) : AnsiString');
23792: //Function RegReadRegisteredOrganizationW( const Access : REGSAM ) : UnicodeString');
23793: Function RegReadRegisteredOwner( const Access : REGSAM ) : string');
23794: Function RegReadRegisteredOwnerA( const Access : REGSAM ) : AnsiString');
23795: //Function RegReadRegisteredOwnerW( const Access : REGSAM ) : UnicodeString');
23796: Function RegReadStrDef( const Key : HKEY; const SubKey : string; const ValueName : string; const Default : string; const Access : REGSAM ) : string');
23797: Function RegReadStrDefA( const Key : HKEY; const SubKey : AnsiString; const ValueName : AnsiString; const Default : AnsiString; const Access : REGSAM ) : AnsiString');
23798: Function StrDecodeUrlA( const Value : RawByteString ) : RawByteString');
23799: Function StrEncodeUrlA( const Value : RawByteString ) : RawByteString');
23800: Function diStrEnd( const s : PChar ) : PChar');
23801: Function StrEndA( const s : PChar ) : PChar');
23802: //Function StrEndW( const s : PWideChar ) : PWideChar );
23803: Procedure StrIncludeTrailingChar( var s : string; const c : Char );
23804: Procedure StrIncludeTrailingCharA( var s : RawByteString; const c : AnsiChar );
23805: //Procedure StrIncludeTrailingCharW( var s : UnicodeString; const c : WideChar );
23806: Function diStrLen( const s : PChar ) : NativeUInt );
23807: Function StrLenA( const s : PChar ) : NativeUInt );
23808: //Function StrLenW( const s : PWideChar ) : NativeUInt );
23809: Function StrRandom( const ASeed : RawByteString; const ACharacters : string; const ALengt : Cardinal ) : string );
23810: Function StrRandomA( const ASeed : RawByteString; const ACharacters : RawByteString; const ALengt : Cardinal ) : RawByteString );
23811: Procedure StrRemoveFromToIA( var Source : RawByteString; const FromString, ToString : RawByteString );
23812: //Procedure StrRemoveFromToIW( var Source : UnicodeString; const FromString, ToString : UnicodeString );
23813: Procedure StrRemoveSpacingA( var s : RawByteString; const SpaceChars : TAnsiCharSet; const ReplaceChar : AnsiChar );
23814: Procedure diStrReplaceChar( var Source : string; const SearchChar, ReplaceChar : Char );
23815: Procedure StrReplaceChar8( var s : Utf8String; const SearchChar, ReplaceChar : AnsiChar );
23816: Procedure StrReplaceCharA( var s : RawByteString; const SearchChar, ReplaceChar : AnsiChar );
23817: //Procedure StrReplaceCharW( var s : UnicodeString; const SearchChar, ReplaceChar : WideChar );
23818: Function diStrReplace( const Source, Search, Replace : string ) : string );
23819: Function StrReplaceA( const Source, Search, Replace : RawByteString ) : RawByteString );
23820: //Function StrReplaceW( const Source, Search, Replace : UnicodeString ) : UnicodeString );
23821: Function StrReplaceI( const Source, Search, Replace : string ) : string );
23822: Function StrReplaceIA( const Source, Search, Replace : RawByteString ) : RawByteString );
23823: //Function StrReplaceIW( const Source, Search, Replace : UnicodeString ) : UnicodeString );
23824: Function StrReplaceLoopA( const Source, Search, Replace : RawByteString ) : RawByteString );
23825: //7Function StrReplaceLoopW( const Source, Search, Replace : UnicodeString ) : UnicodeString );
23826: Function StrReplaceLoopIA( const Source, Search, Replace : RawByteString ) : RawByteString );
23827: //Function StrReplaceLoopIW( const Source, Search, Replace : UnicodeString ) : UnicodeString );
23828: Function RightMostBit( const Value : Cardinal ) : ShortInt );
23829: Function RightMostBit2( const Value : UInt64 ) : ShortInt );
23830: Function LoadStrFromFile( const FileName : string; var s : RawByteString ) : Boolean );
23831: Function FileToStr( const FileName : string; var s : string ) : Boolean );
23832: Function LoadStrAFFromFile( const FileName : AnsiString; var s : RawByteString ) : Boolean );
23833: //Function LoadStrAFFromFileW( const FileName : UnicodeString; var s : RawByteString ) : Boolean );
23834: Function LoadStrWFromFile28( const FileName : string; var s : UnicodeString ) : Boolean );
23835: Function LoadStrWFromFileA( const FileName : Ansistring; var s : UnicodeString ) : Boolean );
23836: //Function LoadStrWFromFileW( const FileName : UnicodeString; var s : UnicodeString ) : Boolean );
23837: Function QuotedStrW( const s : UnicodeString; const Quote : WideChar ) : UnicodeString );

```

```

23838: Function SaveStrToFile( const s : string; const FileName : string ) : Boolean';
23839: Function StrToFile( const s : string; const FileName : string ) : Boolean';
23840: Function SaveStrAToFile( const s : RawByteString; const FileName : string ) : Boolean';
23841: Function SaveStrAToFileA( const s : RawByteString; const FileName : AnsiString ) : Boolean';
23842: //Function SaveStrAToFileW( const s : RawByteString; const FileName : UnicodeString ) : Boolean';
23843: Function SaveStrWToFile( const s : UnicodeString; const FileName : string ) : Boolean';
23844: Function SaveStrWToFileA( const s : UnicodeString; const FileName : AnsiString ) : Boolean';
23845: //Function SaveStrWToFileW( const s : UnicodeString; const FileName : UnicodeString ) : Boolean';
23846: Function StrPosChar( const Source : string; const c : Char; const Start : Cardinal ) : Cardinal';
23847: Function StrPosCharA( const Source : RawByteString; const c : AnsiChar; const Start : Cardinal ) : Cardinal';
23848: //Function StrPosCharW( const Source : UnicodeString; const c : WideChar; const Start : Cardinal ) : Cardinal';
23849: Function StrPosCharBack( const Source : string; const c : Char; const Start : Cardinal ) : Cardinal';
23850: Function StrPosCharBackA( const Source : RawByteString; const c : AnsiChar; const Start : Cardinal ) : Cardinal';
23851: Function StrPosCharsA( const Source:RawByteString; const Search:TAnsiCharSet; const Start:Cardinal): Cardinal';
23852: Function StrPosCharsBackA( const Source:RawByteString;const Search:TAnsiCharSet;const Start:Cardinal): Cardinal';
23853: Function StrPosNotCharsA( const Source:RawByteString;const Search:TAnsiCharSet;const Start:Cardinal): Cardinal';
23854: Function StrPosNotCharsBackA( const Source:RawByteString;const Search:TAnsiCharSet; const Start: Cardinal ) : Cardinal';
23855: Function SetFileDate( const FileHandle : THandle; const Year : Integer; const Month, Day : Word ) : Boolean';
23856: Function SetFileDate2( const FileName : string; const JulianDate : TJJulianDate ) : Boolean';
23857: Function SetFileDateA( const FileName : AnsiString; const JulianDate : TJJulianDate ) : Boolean';
23858: //Function SetFileDateW( const FileName : UnicodeString; const JulianDate : TJJulianDate ) : Boolean';
23859: Function SetFileDateYmd( const FileName : string; const Year : Integer; const Month, Day : Word ) : Boolean';
23860: Function SetFileDateYmdA( const FileName : AnsiString; const Year : Integer; const Month, Day : Word ) : Boolean';
23861: Function SetFileDateYmdW( const FileName:UnicodeString;const Year:Integer; const Month,Day : Word): Boolean';
23862: Function StrContainsChar( const s : string; const c : Char; const Start : Cardinal ) : Boolean';
23863: Function StrContainsCharA( const s : RawByteString; const c : AnsiChar; const Start : Cardinal ) : Boolean';
23864: //Function StrContainsCharW( const s : UnicodeString; const c : WideChar; const Start : Cardinal ) : Boolean';
23865: Function StrContainsCharsA( const s:RawByteString;const Chars:TAnsiCharSet;const Start: Cardinal) : Boolean';
23866: Function diStrSame( const s1, s2 : string ) : Boolean';
23867: Function StrSameA( const s1, s2 : RawByteString ) : Boolean';
23868: //Function StrSameW( const s1, s2 : UnicodeString ) : Boolean';
23869: Function StrSameI( const s1, s2 : string ) : Boolean';
23870: Function StrSameIA( const s1, s2 : RawByteString ) : Boolean';
23871: //Function StrSameIW( const s1, s2 : UnicodeString ) : Boolean';
23872: Function StrSameStart( const s1, s2 : string ) : Boolean';
23873: Function StrSameStartA( const s1, s2 : RawByteString ) : Boolean';
23874: //Function StrSameStartW( const s1, s2 : UnicodeString ) : Boolean';
23875: Function StrSameStartI( const s1, s2 : string ) : Boolean';
23876: Function StrSameStartIA( const s1, s2 : RawByteString ) : Boolean';
23877: //Function StrSameStartIW( const s1, s2 : UnicodeString ) : Boolean';
23878: Function diStrComp( const s1, s2 : string ) : Integer';
23879: Function StrCompA( const s1, s2 : RawByteString ) : Integer';
23880: //Function StrCompW( const s1, s2 : UnicodeString ) : Integer';
23881: Function StrCompI( const s1, s2 : string ) : Integer';
23882: Function StrCompIA( const s1, s2 : RawByteString ) : Integer';
23883: //Function StrCompiW( const s1, s2 : UnicodeString ) : Integer';
23884: Function StrCompNum( const s1, s2 : string ) : Integer';
23885: Function StrCompNumA( const s1, s2 : RawByteString ) : Integer';
23886: // Function StrCompNumW( const s1, s2 : UnicodeString ) : Integer';
23887: Function StrCompNumI( const s1, s2 : string ) : Integer';
23888: Function StrCompNumIA( const s1, s2 : RawByteString ) : Integer';
23889: // Function StrCompNumIW( const s1, s2 : UnicodeString ) : Integer';
23890: Function StrContains( const Search, Source : string; const Start : Cardinal ) : Boolean';
23891: Function StrContainsA( const Search, Source : RawByteString; const Start : Cardinal ) : Boolean';
23892: //Function StrContainsW( const ASearch, ASource : UnicodeString; const AStartPos : Cardinal ) : Boolean';
23893: Function StrContainsI( const Search, Source : string; const Start : Cardinal ) : Boolean';
23894: Function StrContainsIA( const Search, Source : RawByteString; const Start : Cardinal ) : Boolean';
23895: //Function StrContainsIW( const ASearch, ASource : UnicodeString; const AStartPos : Cardinal ) : Boolean';
23896: Function StrCountChar( const ASource : string; const c : Char; const AStartIdx : Cardinal ) : Cardinal';
23897: Function StrCountCharA( const ASource:RawByteString; const c : AnsiChar; const AStartIdx:Cardinal): Cardinal';
23898: Function StrMatchesA( const Search, Source : RawByteString; const AStartIdx : Cardinal ) : Boolean';
23899: Function StrMatchesIA( const Search, Source : RawByteString; const AStartIdx : Cardinal ) : Boolean';
23900: Function StrMatchWild( const Source, Mask : string; const WildChar : Char; const MaskChar : Char ) : Boolean';
23901: Function StrMatchWildA( const Source,Mask:RawByteString;const WildChar:AnsiChar;const MaskChar:AnsiChar):Boolean;
23902: Function StrMatchWildI( const Source, Mask : string; const WildChar : Char; const MaskChar : Char ) : Boolean';
23903: Function StrMatchWildIA( const Source,Mask:RawByteString;const WildChar:AnsiChar;const MaskChar:AnsiChar):Boolean;
23904: Function diStrPos( const ASearch, ASource : string; const AStartPos : Cardinal ) : Cardinal';
23905: Function StrPosA( const ASearch, ASource : RawByteString; const AStartPos : Cardinal ) : Cardinal';
23906: //Function StrPosW( const ASearch, ASource : UnicodeString; const AStartPos : Cardinal ) : Cardinal';
23907: Function StrPosI( const ASearch, ASource : string; const AStartPos : Cardinal ) : Cardinal');

```

```

23908: Function StrPosIA( const ASearch, ASource : RawByteString; const AStartPos : Cardinal ) : Cardinal';
23909: //Function StrPosIW( const ASearch, ASource : UnicodeString; const AStartPos : Cardinal ) : Cardinal';
23910: Function StrPosBackA( const ASearch, ASource : RawByteString; AStart : Cardinal ) : Cardinal';
23911: Function StrPosBackIA( const ASearch, ASource : RawByteString; AStart : Cardinal ) : Cardinal';
23912: //Function StrToIntDefW( const w : UnicodeString; const Default : Integer ) : Integer';
23913: Function StrToInt64DefW( const w : UnicodeString; const Default : Int64 ) : Int64';
23914: Function StrToUpper( const s : string ) : string';
23915: Function StrToUpperA( const s : RawByteString ) : RawByteString';
23916: //Function StrToUpperW( const s : UnicodeString ) : UnicodeString';
23917: Procedure StrToUpperInPlace( var s : string );
23918: Procedure StrToUpperInPlaceA( var s : AnsiString );
23919: //Procedure StrToUpperInPlaceW31( var s : WideString );
23920: //Procedure StrToUpperInPlaceW32( var s : UnicodeString );
23921: Function StrToLower( const s : string ) : string';
23922: Function StrToLowerA( const s : RawByteString ) : RawByteString';
23923: //Function StrToLowerW( const s : UnicodeString ) : UnicodeString';
23924: Procedure StrToLowerInPlace( var s : string );
23925: Procedure StrToLowerInPlaceA( var s : AnsiString );
23926: //Procedure StrToLowerInPlaceW33( var s : WideString );
23927: //Procedure StrToLowerInPlaceW34( var s : UnicodeString );
23928: Procedure StrTrimUriFragmentA( var Value : RawByteString );
23929: //Procedure StrTrimUriFragmentW( var Value : UnicodeString );
23930: //Function StrExtractUriFragmentW( var Value : UnicodeString ) : UnicodeString';
23931: Function StrCountUtf8Chars( const AValue: Utf8String ) : Cardinal';
23932: Function StrDecodeUtf8( const AValue: Utf8String ) : UnicodeString';
23933: Function StrEncodeUtf8( const AValue: UnicodeString ) : Utf8String';
23934: Function diSysErrorMessage( const MessageID : Cardinal ) : string';
23935: Function SysErrorMessageA( const MessageID: Cardinal ) : AnsiString';
23936: //Function SysErrorMessageW( const MessageID : Cardinal ) : UnicodeString';
23937: Function TextExtentW( const DC : HDC; const Text : UnicodeString ) : TSize';
23938: Function TextHeightW( const DC : HDC; const Text : UnicodeString ) : Integer';
23939: Function TextWidthW( const DC : HDC; const Text : UnicodeString ) : Integer';
23940: Function diStrTrim( const Source : string ) : string';
23941: Function StrTrimA( const Source : RawByteString ) : RawByteString';
23942: //Function StrTrimW( const w : UnicodeString ) : UnicodeString';
23943: Function StrTrimCharA( const Source : RawByteString; const CharToTrim : AnsiChar ) : RawByteString';
23944: Function StrTrimCharsA( const Source : RawByteString; const CharsToTrim : TAnsiCharSet ) : RawByteString';
23945: //Function StrTrimCharsW( const s : UnicodeString; const IsCharToTrim : TValidateCharFuncW ) : UnicodeString';
23946: Procedure TrimLeftByRefA( var s : RawByteString; const Chars : TAnsiCharSet );
23947: Function TrimRightA( const Source : RawByteString; const s : TAnsiCharSet ) : RawByteString';
23948: Procedure TrimRightByRefA( var Source : RawByteString; const s : TAnsiCharSet );
23949: Procedure StrTrimCompressA( var s:RawByteString;const TrimCompressChars:TAnsiCharSet;const ReplaceChar:AnsiChar );
23950: Function TryStrToIntW( const w : UnicodeString; out Value : Integer ) : Boolean';
23951: Function TryStrToInt64W( const w : UnicodeString; out Value : Int64 ) : Boolean';
23952: Function ValInt( const p : PChar; const l : Integer; out Code : Integer ) : Integer';
23953: Function ValIntA36( p : PChar; l : Integer; out Code : Integer ) : Integer';
23954: //Function ValIntW37( p : PWideChar; l : Integer; out Code : Integer ) : Integer';
23955: Function ValInt2( const s : string; out Code : Integer ) : Integer';
23956: Function ValIntA39( const s : RawByteString; out Code : Integer ) : Integer';
23957: Function ValIntW40( const s : UnicodeString; out Code : Integer ) : Integer';
23958: Function ValInt64A41( p : PChar; l : Integer; out Code : Integer ) : Int64';
23959: //Function ValInt64W42( p : PWideChar; l : Integer; out Code : Integer ) : Int64';
23960: Function ValInt64A43( const s : RawByteString; out Code : Integer ) : Int64';
23961: Function ValInt64W44( const s : UnicodeString; out Code : Integer ) : Int64';
23962: Function YmdToIsoDateStr( const Year : Integer; const Month : Word; const Day : Word ) : string';
23963: Function YmdToIsoDateStrA( const Year : Integer; const Month : Word; const Day : Word ) : RawByteString';
23964: Function YmdToIsoDateStrW( const Year : Integer; const Month : Word; const Day : Word ) : UnicodeString';
23965: Function CharIsLetterW( const c : WideChar ) : Boolean';
23966: Function CharIsLetterCommonW( const c : WideChar ) : Boolean';
23967: Function CharIsLetterUpperCaseW( const c : WideChar ) : Boolean';
23968: Function CharIsLetterLowerCaseW( const c : WideChar ) : Boolean';
23969: Function CharIsLetterTitleCaseW( const c : WideChar ) : Boolean';
23970: Function CharIsLetterModifierW( const c : WideChar ) : Boolean';
23971: Function CharIsLetterOtherW( const c : WideChar ) : Boolean';
23972: Function CharIsMarkW( const c : WideChar ) : Boolean';
23973: Function CharIsMarkNon_SpacingW( const c : WideChar ) : Boolean';
23974: Function CharIsMarkSpacing_CombinedW( const c : WideChar ) : Boolean';
23975: Function CharIsMarkEnclosingW( const c : WideChar ) : Boolean';
23976: Function CharIsNumberW( const c : WideChar ) : Boolean';
23977: Function CharIsNumber_DecimalW( const c : WideChar ) : Boolean';
23978: Function CharIsNumber_LetterW( const c : WideChar ) : Boolean';
23979: Function CharIsNumber_OtherW( const c : WideChar ) : Boolean';
23980: Function CharIsPunctuationW( const c : WideChar ) : Boolean';
23981: Function CharIsPunctuation_ConnectorW( const c : WideChar ) : Boolean';
23982: Function CharIsPunctuation_DashW( const c : WideChar ) : Boolean';
23983: Function CharIsPunctuation_OpenW( const c : WideChar ) : Boolean';
23984: Function CharIsPunctuation_CloseW( const c : WideChar ) : Boolean';
23985: Function CharIsPunctuation_InitialQuoteW( const c : WideChar ) : Boolean';
23986: Function CharIsPunctuation_FinalQuoteW( const c : WideChar ) : Boolean';
23987: Function CharIsPunctuation_OtherW( const c : WideChar ) : Boolean';
23988: Function CharIsSymbolW( const c : WideChar ) : Boolean';
23989: Function CharIsSymbolMathW( const c : WideChar ) : Boolean';
23990: Function CharIsSymbolCurrencyW( const c : WideChar ) : Boolean';
23991: Function CharIsSymbolModifierW( const c : WideChar ) : Boolean';
23992: Function CharIsSymbolOtherW( const c : WideChar ) : Boolean';
23993: Function CharIsSeparatorW( const c : WideChar ) : Boolean';
23994: Function CharIsSeparatorSpaceW( const c : WideChar ) : Boolean';

```

```

23995: Function CharIsSeparatorLineW( const c : WideChar ) : Boolean';
23996: Function CharIsSeparatorParagraphW( const c : WideChar ) : Boolean';
23997: Function CharIsOtherW( const c : WideChar ) : Boolean';
23998: Function CharIsOtherControlW( const c : WideChar ) : Boolean';
23999: Function CharIsOtherFormatW( const c : WideChar ) : Boolean';
24000: Function CharIsOtherSurrogateW( const c : WideChar ) : Boolean';
24001: Function CharIsOtherPrivateUseW( const c : WideChar ) : Boolean';
24002: Function BitClear( const Bits, BitNo : Integer ) : Integer';
24003: Function BitSet( const Bits, BitIndex : Integer ) : Integer';
24004: Function BitSetTo( const Bits, BitIndex : Integer; const Value : Boolean ) : Integer';
24005: Function BitTest( const Bits, BitIndex : Integer ) : Boolean';
24006: Function CharCanonicalCombiningClassW( const Char : WideChar ) : Cardinal';
24007: Function CharIsAlphaW( const c : WideChar ) : Boolean';
24008: Function CharIsAlphaNumW( const c : WideChar ) : Boolean';
24009: Function CharIsCrLf( const c : Char ) : Boolean';
24010: Function CharIsCrLfA( const c : AnsiChar ) : Boolean';
24011: //Function CharIsCrLfW( const c : WideChar ) : Boolean';
24012: Function diCharIsDigit( const c : Char ) : Boolean';
24013: Function CharIsDigitA( const c : AnsiChar ) : Boolean';
24014: //Function CharIsDigitW( const c : WideChar ) : Boolean';
24015: Function CharIsHangulW( const Char : WideChar ) : Boolean';
24016: Function CharIsHexDigitW( const c : WideChar ) : Boolean';
24017: Function CharIsWhiteSpaceW( const c : WideChar ) : Boolean';
24018: Function CharToCaseFoldW( const Char : WideChar ) : WideChar';
24019: Function CharToLowerW( const Char : WideChar ) : WideChar';
24020: Function CharToUpperW( const Char : WideChar ) : WideChar';
24021: Function CharToTitleW( const Char : WideChar ) : WideChar';
24022: Function DayOfJulianDate( const JulianDate : TJ JulianDate ) : Word';
24023: Function diDayOfWeek( const JulianDate : TJ JulianDate ) : Word';
24024: Function DayOfWeekYmd( const Year : Integer; const Month, Day : Word ) : Word';
24025: Function diDaysInMonth( const JulianDate : TJ JulianDate ) : Word';
24026: Function DaysInMonthYm( const Year : Integer; const Month : Word ) : Word';
24027: Procedure DecDay( var Year : Integer; var Month, Day : Word );
24028: Procedure DecDays( var Year : Integer; var Month, Day : Word; const Days : Integer );
24029: Function diDeleteDirectory( const Dir : string; const DeleteItself : Boolean ) : Boolean';
24030: Function DeleteDirectoryA( Dir : AnsiString; const DeleteItself : Boolean ) : Boolean';
24031: //Function DeleteDirectoryW( Dir : UnicodeString; const DeleteItself : Boolean ) : Boolean';
24032: Function diEasterSunday( const Year : Integer ) : TJ JulianDate';
24033: Procedure EasterSundayYmd( const Year : Integer; out Month, Day : Word );
24034: Function diFirstDayOfWeek( const JulianDate : TJ JulianDate ) : TJ JulianDate';
24035: Procedure FirstDayOfWeekYmd( var Year : Integer; var Month, Day : Word );
24036: Function diFirstDayOfMonth( const Julian : TJ JulianDate ) : TJ JulianDate';
24037: Procedure FirstDayOfMonthYmd( const Year : Integer; const Month : Word; out Day : Word );
24038: Function diForceDirectories( const Dir : string ) : Boolean';
24039: Function ForceDirectoriesA( Dir : AnsiString ) : Boolean';
24040: //Function ForceDirectoriesW( Dir : UnicodeString ) : Boolean';
24041: Procedure FreeMemAndNil( var Ptr : TObject );
24042: Function diGetCurrentFolder : string';
24043: Function GetCurrentFolderA( AnsiString );
24044: //Function GetCurrentFolderW : UnicodeString';
24045: Procedure SetCurrentFolder( const NewFolder : string );
24046: Procedure SetCurrentFolderA( const NewFolder : AnsiString );
24047: //Procedure SetCurrentFolderW( const NewFolder : UnicodeString );
24048: Function diGetDesktopFolder : string';
24049: Function GetDesktopFolderA( AnsiString );
24050: //Function GetDesktopFolderW : UnicodeString';
24051: Function diGetFileSize( const AFileName : string ) : Int64';
24052: Function GetFileSizeA( const AFileName : AnsiString ) : Int64';
24053: //Function GetFileSizeW( const AFileName : UnicodeString ) : Int64';
24054: Function diGetDesktopDirectoryFolder : string';
24055: Function GetDesktopDirectoryFolderA( AnsiString );
24056: //Function GetDesktopDirectoryFolderW : UnicodeString';
24057: Function GetFileLastWriteTime( const FileName : string; out FileTime : TFileTime ) : Boolean';
24058: Function GetFileLastWriteTimeA( const FileName : AnsiString; out FileTime : TFileTime ) : Boolean';
24059: //Function GetFileLastWriteTimeW( const FileName : UnicodeString; out FileTime : TFileTime ) : Boolean';
24060: Function diGetPersonalFolder( const PersonalFolder : Integer ) : string';
24061: Function GetPersonalFolderA( AnsiString );
24062: //Function GetPersonalFolderW : UnicodeString';
24063: Function GetSpecialFolder( const SpecialFolder : Integer ) : string';
24064: Function GetSpecialFolderA( const SpecialFolder : Integer ) : AnsiString';
24065: //Function GetSpecialFolderW( const SpecialFolder : Integer ) : UnicodeString';
24066: Procedure diIncMonth( var Year : Integer; var Month, Day : Word );
24067: Procedure diIncMonths( var Year : Integer; var Month, Day : Word; const NumberOfMonths : Integer );
24068: Procedure diIncDay( var Year : Integer; var Month, Day : Word );
24069: Procedure IncDays( var Year : Integer; var Month, Day : Word; const Days : Integer );
24070: Function IsDateValid( const Year : Integer; const Month, Day : Word ) : Boolean';
24071: Function IsHolidayInGermany( const Julian : TJ JulianDate ) : Boolean';
24072: Function IsHolidayInGermanyYmd( const Year : Integer; const Month, Day : Word ) : Boolean';
24073: Function diIsLeapYear( const Year : Integer ) : Boolean';
24074: Function ISODateToJulianDate( const ISODate : TIsoDate ) : TJ JulianDate';
24075: Procedure ISODateToYmd( const ISODate : TIsoDate; out Year : Integer; out Month, Day : Word );
24076: Function IsCharLowLineW( const c : WideChar ) : Boolean';
24077: Function IsCharQuoteW( const c : WideChar ) : Boolean';
24078: Function IsShiftKeyDown : Boolean';
24079: Function IsCharWhiteSpaceOrAmpersandW( const c : WideChar ) : Boolean';
24080: Function IsCharWhiteSpaceOrNoBreakSpaceW( const c : WideChar ) : Boolean';
24081: Function IsCharWhiteSpaceOrColonW( const c : WideChar ) : Boolean';
24082: Function CharIsWhiteSpaceGtW( const c : WideChar ) : Boolean';
24083: Function CharIsWhiteSpaceLtW( const c : WideChar ) : Boolean';

```

```

24084: Function CharIsWhiteSpaceHyphenW( const c : WideChar ) : Boolean';
24085: Function CharIsWhiteSpaceHyphenGtW( const c : WideChar ) : Boolean';
24086: Function IsCharWordSeparatorW( const c : WideChar ) : Boolean';
24087: Function diISOWeekNumber( const JulianDate : TJJulianDate ) : Word';
24088: Function ISOWeekNumberYmd( const Year : Integer; const Month, Day : Word ) : Word';
24089: Function ISOWeekToJulianDate( const Year : Integer; const WeekOfYear : Word; const DayOfWeek : Word ) : TJJulianDate );
24090: Function JulianDateIsWeekDay( const JulianDate : TJJulianDate ) : Boolean';
24091: Function JulianDateToIsoDate( const Julian : TJJulianDate ) : TIsoDate');
24092: Procedure JulianDateToYmd( const JulianDate : TJJulianDate; out Year : Integer; out Month, Day : Word );
24093: Function LastDayOfMonth( const JulianDate : TJJulianDate ) : TJJulianDate');
24094: Procedure LastDayOfMonthYmd( const Year : Integer; const Month : Word; out Day : Word );
24095: Function LastDayOfWeek( const JulianDate : TJJulianDate ) : TJJulianDate');
24096: Procedure LastDayOfWeekYmd( var Year : Integer; var Month, Day : Word );
24097: Function LastSysErrorMessage : string');
24098: Function LastSysErrorMessageA : AnsiString');
24099: Function LastSysErrorMessageW : UnicodeString');
24100: Function diMax( const a : Integer; const b : Integer ) : Integer');
24101: Function diMax3( const a, b, c : Integer ) : Integer');
24102: Function MaxCard( const a : Cardinal; const b : Cardinal ) : Cardinal');
24103: Function MaxCard3( const a : Cardinal; const b : Cardinal; const c : Cardinal ) : Cardinal');
24104: Function diMaxInt64( const a : Int64; const b : Int64 ) : Int64');
24105: Function diMaxInt64( const a : Int64; const b : Int64; const c : Int64 ) : Int64');
24106: Function diMin( const a, b : Integer ) : Integer');
24107: Function diMin3( const a, b, c : Integer ) : Integer');
24108: Function MinCard( const a, b : Cardinal ) : Cardinal');
24109: Function Mincard3( const a, b, c : Cardinal ) : Cardinal');
24110: Function diMinInt64( const a, b : Int64 ) : Int64');
24111: Function diMinInt64( const a, b, c : Int64 ) : Int64');
24112: Function diMinInt64U( const a, b : UInt64 ) : UInt64');
24113: Function diMinInt64U( const a, b, c : UInt64 ) : UInt64');
24114: Function MonthOfJulianDate( const JulianDate : TJJulianDate ) : Word');
24115: Function YearOfJulianDate( const JulianDate : TJJulianDate ) : Integer');
24116: Function YmdToIsoDate( const Year : Integer; const Month, Day : Word ) : TIsoDate');
24117: Function YmdToJulianDate( const Year : Integer; const Month, Day : Word ) : TJJulianDate');
24118: end;
24119:
24120: TDLLVersionInfo=Record
24121:   cbSize, // Size of the structure, in bytes.
24122:   dwMajorVersion, // Major version of the DLL
24123:   dwMinorVersion, // Minor version of the DLL
24124:   dwBuildNumber, // Build number of the DLL
24125:   dwPlatformID: DWord; // Identifies the platform for which the DLL was built
24126: end;
24127:
24128: command1:= 'play "'+songpath+'maxbox.wav"'; command2:= 'play "'+songpath+'moon.wav"';
24129: SendMCICmd('open waveaudio shareable'); //parallels
24130: SendMCICmd('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\maxbox.wav"');
24131: SendMCICmd('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\moon.wav"');
24132: SendMCICmd('close waveaudio');
24133:
24134:
24135: /////////////////////////////////
24136: All maxbox Tutorials Table of Content 2014/2015
24137: /////////////////////////////////
24138: Tutorial 00 Function-Coding (Blix the Programmer)
24139: Tutorial 01 Procedural-Coding
24140: Tutorial 02 OO-Programming
24141: Tutorial 03 Modular Coding
24142: Tutorial 04 UML Use Case Coding
24143: Tutorial 05 Internet Coding
24144: Tutorial 06 Network Coding
24145: Tutorial 07 Game Graphics Coding
24146: Tutorial 08 Operating System Coding
24147: Tutorial 09 Database Coding
24148: Tutorial 10 Statistic Coding
24149: Tutorial 11 Forms Coding
24150: Tutorial 12 SQL DB Coding
24151: Tutorial 13 Crypto Coding
24152: Tutorial 14 Parallel Coding
24153: Tutorial 15 Serial RS232 Coding
24154: Tutorial 16 Event Driven Coding
24155: Tutorial 17 Web Server Coding
24156: Tutorial 18 Arduino System Coding
24157: Tutorial 18_3 RGB LED System Coding
24158: Tutorial 19 WinCOM /Arduino Coding
24159: Tutorial 20 Regular Expressions RegEx
24160: Tutorial 21 Android Coding (coming 2013)
24161: Tutorial 22 Services Programming
24162: Tutorial 23 Real Time Systems
24163: Tutorial 24 Clean Code
24164: Tutorial 25 maxbox Configuration I+II
24165: Tutorial 26 Socket Programming with TCP
24166: Tutorial 27 XML & TreeView
24167: Tutorial 28 DLL Coding (available)
24168: Tutorial 29 UML Scripting (2014)
24169: Tutorial 30 Web of Things (2014)
24170: Tutorial 31 Closures (2014)
24171: Tutorial 32 SQL Firebird (2014)
24172: Tutorial 33 Oscilloscope (available)

```

```

24173: Tutorial 34 GPS Navigation (2014)
24174: Tutorial 35 Web Box (available)
24175: Tutorial 36 Unit Testing (avail)
24176: Tutorial 37 API Coding (avail)
24177: Tutorial 38 3D Coding (coming 2015)
24178: Tutorial 39 GEO Map Coding (available)
24179: Tutorial 39_1 GEO Map Layers Coding (available)
24180: Tutorial 40 REST Coding (coming 2015)
24181: Tutorial 41 Big Numbers Coding (coming 2015)
24182: Tutorial 42 Parallel Processing (coming 2015)
24183:
24184:
24185: Doc ref Docu for all Type Class and Const in maxbox_types.pdf
24186: using Docu for this file is maxbox_functions_all.pdf
24187: PEP - Pascal Education Program Low Lib Lab ShellHell in UDEMY
24188:
24189: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials
24190: http://stackoverflow.com/tags/pascalscript/hot
24191: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
24192: http://sourceforge.net/projects/maxbox #locs:51620
24193: http://sourceforge.net/apps/mediawiki/maxbox
24194: http://www.blaisepascal.eu/
24195: https://github.com/maxkleiner/maxbox3.git
24196: http://www.heise.de/download/maxbox-1176464.html
24197: http://www.softpedia.com/get/Programming/Other-Programming-Files/maxbox.shtml
24198: https://www.facebook.com/pages/Programming-maxBox/166844836691703
24199: http://www.softwareschule.ch/arduino_training.pdf
24200: http://www.delphiarea.com
24201: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
24202: http://entwickler-konferenz.de/2014/speakers/max-kleiner
24203: http://www.heise.de/download/maxbox-1176464.html
24204: https://www.udemy.com/learn-coding-from-the-scratch
24205: http://www.slideshare.net/maxkleiner1/codesign-2015
24206: https://www.dropbox.com/s/yolconwmq4oqta4/Blaise_2and3_SP_Total.pdf?dl=0
24207: http://www.softwareschule.ch/download/maxbox_promo.png
24208: http://max.kleiner.com/maxbox_functions_all.htm
24209: http://max.kleiner.com/boxart.htm
24210: http://www.jurgott.org/linkage/util.htm
24211: http://www.swissdelphicenter.ch/en/niklauswirth.php
24212: http://www.softwareschule.ch/images/maxbox_20years_delphi.jpg
24213:
24214:
24215: All maxbox Examples List
24216: https://github.com/maxkleiner/maxbox3/releases
24217: ****
24218: 000_pas_baseconvert.txt
24219: 000_pas_baseconvert.txt_encrypt
24220: 000_pas_baseconvert.txt_decrypt
24221: 001_1_pas_functest - Kopie.txt
24222: 001_1_pas_functest.txt
24223: 001_1_pas_functest2.txt
24224: 001_1_pas_functest_clx2.txt
24225: 001_1_pas_functest_clx2_2.txt
24226: 001_1_pas_functest_openarray.txt
24227: 001_pas_lottogen.txt
24228: 001_pas_lottogen_template.txt
24229: 001_pas_lottogen_txtcopy
24230: 002_pas_russianroulette.txt
24231: 002_pas_russianroulette.txtcopy
24232: 002_pas_russianroulette.txtcopy_decrypt
24233: 002_pas_russianroulette.txtcopy_encrypt
24234: 003_pas_motion.txt
24235: 003_pas_motion.txtcopy
24236: 004_pas_search.txt
24237: 004_pas_search_replace.txt
24238: 004_search_replace_allfunctionlist.txt
24239: 005_pas_oodesign.txt
24240: 005_pas_shelllink.txt
24241: 006_pas_oobatch.txt
24242: 007_pas_streamcopy.txt
24243: 008_EINMALEINS_FUNC.TXT
24244: 008_explanation.txt
24245: 008_pas_verwechselt.txt
24246: 008_pas_verwechselt_ibz_bern_func.txt
24247: 008_stack_ibz.TXT
24248: 009_pas_umrunner.txt
24249: 009_pas_umrunner_all.txt
24250: 009_pas_umrunner_componenttest.txt
24251: 009_pas_umrunner_solution.txt
24252: 009_pas_umrunner_solution_2step.txt
24253: 010_pas_oodesign_solution.txt
24254: 011_pas_puzzlepas_defect.txt
24255: 012_pas_umrunner_solution.txt
24256: 012_pas_umrunner_solution2.txt
24257: 013_pas_linenumber.txt
24258: 014_pas_primetest.txt
24259: 014_pas_primetest_first.txt
24260: 014_pas_primetest_sync.txt
24261: 015_pas_designbycontract.txt
282_fadengraphik.txt
283_SQL_API_messagetimeout.txt
284_SysTools4.txt
285_MineForm_GR32.TXT
285_MineForm_GR32main.TXT
285_MineForm_GR32mainsolution.TXT
285_MineForm_propas.TXT
285_MineForm_propas2.TXT
285_minesweeper2.TXT
285_Patterns_process.txt
286_colormixer_jpeg_charcounter.txt
286_colormixer_jpeg_charcounter2.txt
287_eventhandling.txt
287_eventhandling2.txt
287_eventhandling2_negpower.txt
288_bitblt.txt
288_bitblt_resize.txt
289_regression.txt
289_regression2.txt
290_bestofbox.txt
290_bestofbox2.txt
290_bestofbox3.txt
291_3sort_visual_thread.txt
292_refactoring2.txt
293_bold_utils.txt
293_ib_utils.txt
293_ib_utils_timetest.txt
294_maxcalc_demo.txt
294_maxcalc_demo2.txt
295_easter_calendar.txt
295_easter_calendar2.txt
295_easter_combobox.txt
297_atomimage.txt
297_atomimage2.txt
297_atomimage3.txt
297_atomimage4.txt
297_maxonmotor.TXT
297_maxon_atomimage9.txt
298_bitblt_animation.txt
298_bitblt_animation2.txt
298_bitblt_animation3.txt
298_bitblt_animation4.txt
298_bitblt_animation4_screensaver.txt
298_bitblt_animation5_screensaver.txt

```

```

24262: 015_pas_designbycontract_solution.txt          299_animation.txt
24263: 016_pas_searchrec.txt                         299_animationmotor_arduino.txt
24264: 017_chartgen.txt                            299_animation_formprototype.txt
24265: 018_data_simulator.txt                      299_realtimeclock_arduino.txt
24266: 019_dez_to_bin.txt                          299_realtimeclock_arduino2.txt
24267: 019_dez_to_bin_grenzwert_ibz.txt           300_treeview.txt
24268: 020_proc_feedback.txt                       300_treeview_test.txt
24269: 021_pas_symkey.txt                         300_treeview_test2.txt
24270: 021_pas_symkey_solution.txt                 300_treeview_test3.txt
24271: 022_pas_filestreams.txt                     301_LED_Arduino3.txt
24272: 023_pas_find_searchrec.txt                  301_led_arduino3_simple.txt
24273: 023_pas_pathfind.txt                       301_led_arduino3_simplecode.txt
24274: 024_pas_TFileStream_records.txt            301_log_arduino.txt
24275: 025_prime_direct.txt                       301_log_arduino2.txt
24276: 026_pas_memorystream.txt                   301_SQL_DBfirebird3.txt
24277: 027_pas_shellexecute_beta.txt              301_SQL_DBfirebird4.txt
24278: 027_pas_shellexecute_solution.txt         302_LCLActivity_java.txt
24279: 028_pas_dataset.txt                        302_LED_DataLogger.txt
24280: 029_pas_assignfile.txt                     303_Android_LCLActivity_java.txt
24281: 029_pas_assignfile_dragndropexe.txt       303_webserver.txt
24282: 030_palindrome_2.txt                       303_webserver2.txt
24283: 030_palindrome_tester.txt                  303_webserver_alldocs2.txt
24284: 030_pas_recursion.txt                     303_webserver_alldocs2_tester.txt
24285: 030_pas_recursion2.txt                    303_webserver_minimal.txt
24286: 031_pas_hashcode.txt                      303_webserver_simple.txt
24287: 032_pas_crc_const.txt                     304_st_system.txt
24288: 033_pas_cipher.txt                        305_indy_elizahttpserver.TXT
24289: 033_pas_cipher_def.txt                   305_indy_elizahttpserver2.TXT
24290: 033_pas_cipher_file_2_solution.txt        305_indy_elizahttpserver3.TXT
24291: 034_pas_soundbox.txt                      305_indy_elizahttpserver4file.TXT
24292: 035_pas_crcscript.txt                     305_webserver_arduino.txt
24293: 035_pas_CRCscript_modbus.txt              305_webserver_arduino2.txt
24294: 036_pas_includetest.txt                  305_webserver_arduino3.txt
24295: 036_pas_includetest_basta.txt             305_webserver_arduino3ibz.txt
24296: 037_pas_define_demo32.txt                305_webserver_arduino3ibz_rgb_led.txt
24297: 038_pas_box_demonstrator.txt              305_webserver_arduino3test.txt
24298: 039_pas_dllcall.txt                       306_SPS_http_command.txt
24299: 040_paspointer.txt                        307_all_booleanlogic.txt
24300: 040_paspointer_old.txt                   308_bitbox3.txt
24301: 041_pasplotter.txt                       308_bitbox3_exec.txt
24302: 041_pasplotter_plus.txt                  308_boolean_animation.txt
24303: 042_pas_kgv_ggt.txt                      308_boolean_animation2.txt
24304: 043_pas_proceduretype.txt                309_regex_power.txt
24305: 044_pas_14queens_solwith14.txt           309_regex_powertester2.txt
24306: 044_pas_8queens.txt                      309_regex_powertester3.txt
24307: 044_pas_8queens_sol2.txt                 310_regex_decorator.TXT
24308: 044_pas_8queens_solutions.txt            312_ListView.txt
24309: 044_queens_performer.txt                 313_dmath_dll.txt
24310: 044_queens_performer2.txt                314_fundamentals4_tester.TXT
24311: 044_queens_performer2tester.txt          315_funcplot_dmath.TXT
24312: 045_pas_listhandling.txt                 316_fileutils_cdatetime_tester.TXT
24313: 046_pas_records.txt                      317_excel_export_tester.TXT
24314: 047_pas_modula10.txt                     318_excel_export.TXT
24315: 048_pas_romans.txt                      318_excel_export2.TXT
24316: 049_pas_ifdemo.txt                       318_excel_export3.TXT
24317: 049_pas_ifdemo_BROKER.txt                318_excel_export3_tester.TXT
24318: 050_pas_primetest2.txt                   319_superfunctions_math.TXT
24319: 050_pas_primetester_thieves.txt          319_superfunctions_mathdefect.TXT
24320: 050_program_starter.txt                  320_superfunctions.TXT
24321: 050_program_starter_performance.txt     320_superfunctions2.TXT
24322: 051_pas_findtext_solution.txt             321_SQL_Excel.txt
24323: 052_pas_text_as_stream.txt               321_SQL_Excel2.txt
24324: 052_pas_text_as_stream_include.txt       321_SQL_Excel_Export.txt
24325: 053_pas_singleton.txt                   321_SQL_ExportExec.txt
24326: 054_pas_speakpassword.txt                321_SQL_ExportTest.txt
24327: 054_pas_speakpassword2.txt               321_SQL_SAS_tester3.txt
24328: 054_pas_speakpassword_searchtest.txt    321_SQL_SAS_tester3_selfcompile.txt
24329: 055_pas_factorylist.txt                 321_SQL_SAS_tester3_selfcompile2.txt
24330: 056_pas_demeter.txt                     321_SQL_SAS_tester4.txt
24331: 057_pas_dirfinder.txt                   321_SQL_SAS_updater.txt
24332: 058_pas_filefinder.txt                  322_timezones.TXT
24333: 058_pas_filefinder_pdf.txt              323_datefind_fulltext_search.txt
24334: 058_pas_filefinder_screview.txt          323_datefind_fulltext_searchtester.txt
24335: 058_pas_filefinder_screview2.txt         324_interfacenavi.TXT
24336: 058_pas_filefinder_screview3.txt         325_ampelsteuerung.txt
24337: 059_pas_timertest.txt                   325_analogclock.txt
24338: 059_pas_timertest_2.txt                 326_world_analogclock.txt
24339: 059_pas_timertest_time_solution.txt      326_world_analogclock2.txt
24340: 059_timerobject_starter2.txt             327_atomimage_clock.txt
24341: 059_timerobject_starter2_ibz2_async.txt   328_starfield.txt
24342: 059_timerobject_starter2_uml.txt         329_starfield2.txt
24343: 059_timerobject_starter2_uml_main.txt     330_myclock.txt
24344: 059_timerobject_starter4_ibz.txt          330_myclock2.txt
24345: 060_pas_datefind.txt                     331_SQL_DBfirebird4.txt
24346: 060_pas_datefind_exceptions2.txt          332_jprofiler.txt
24347: 060_pas_datefind_exceptions_CHECKTEST.txt 332_jprofiler_form.txt
24348: 060_pas_datefind_fulltext.txt              332_jprofiler_form2.txt
24349: 060_pas_datefind_plus.txt                 333_querybyexample.txt
24350: 060_pas_datefind_plus_mydate.txt          333_querybyexample2.txt

```

```

24351: 061_pas_randomwalk.txt
24352: 061_pas_randomwalk_plus.txt
24353: 062_paskorrelation.txt
24354: 063_pas_calculateform.txt
24355: 063_pas_calculateform_2list.txt
24356: 064_pas_timetest.txt
24357: 065_pas_bitcounter.txt
24358: 066_pas_eliza.txt
24359: 066_pas_eliza_include_sol.txt
24360: 067_pas_morse.txt
24361: 068_pas_piezo_sound.txt
24362: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
24363: 069_my_LEDBOX.TXT
24364: 069_pas_ledmatrix.txt
24365: 069_pas_LEDMATRIX_Alphabet.txt
24366: 069_pas_LEDMATRIX_Alphabet_run.txt
24367: 069_pas_LEDMATRIX_Alphabet_tester.txt
24368: 069_PAS_LEDMATRIX_COLOR.TXT
24369: 069_pas_ledmatrix_fixedit.txt
24370: 069_pas_LEDMATRIX_soundbox.txt
24371: 069_pas_LEDMATRIX_soundbox2.txt
24372: 069_Richter_MATRIX.TXT
24373: 070_pas_functionplot.txt
24374: 070_pas_functionplotter2.txt
24375: 070_pas_functionplotter2_mx4.txt
24376: 070_pas_functionplotter2_tester.txt
24377: 070_pas_functionplotter3.txt
24378: 070_pas_functionplotter4.txt
24379: 070_pas_functionplotter_digital.txt
24380: 070_pas_functionplotter_elliptic.txt
24381: 070_pas_function_helmholtz.txt
24382: 070_pas_textcheck_experimental.txt
24383: 071_pas_graphics.txt
24384: 071_pas_graphics_drawsym.txt
24385: 071_pas_graphics_drawsym_save.txt
24386: 071_pas_graphics_random.txt
24387: 072_pas_fractals.txt
24388: 072_pas_fractals_2.txt
24389: 072_pas_fractals_blackhole.txt
24390: 072_pas_fractals_performance.txt
24391: 072_pas_fractals_performance_new.txt
24392: 072_pas_fractals_performance_sharp.txt
24393: 072_pas_fractals_performance.txt
24394: 072_pas_fractals_performance_mx4.txt
24395: 073_pas_forms.txt
24396: 074_pas_chartgenerator.txt
24397: 074_pas_chartgenerator_solution.txt
24398: 074_pas_chartgenerator_solution_back.txt
24399: 074_pas_charts.txt
24400: 075_bitmap_Artwork2.txt
24401: 075_pas_bitmappuzzle.txt
24402: 075_pas_bitmappuzzle24.prod.txt
24403: 075_pas_bitmappuzzle2.prod.txt
24404: 075_pas_bitmappuzzle3.txt
24405: 075_pas_bitmapsolve.txt
24406: 075_pas_bitmap_Artwork.txt
24407: 075_pas_puzzlesolution.txt
24408: 076_pas_3dcube.txt
24409: 076_pas_circle.txt
24410: 077_pas_mmshow.txt
24411: 078_pas_pi.txt
24412: 079_pas_3dcube_animation.txt
24413: 079_pas_3dcube_animation4.txt
24414: 079_pas_3dcube_plus.txt
24415: 080_pas_hanoi.txt
24416: 080_pas_hanoi2.txt
24417: 080_pas_hanoi2_file.txt
24418: 080_pas_hanoi2_sol.txt
24419: 080_pas_hanoi2_tester.txt
24420: 080_pas_hanoi2_tester_fast.txt
24421: 080_pas_hanoi3.txt
24422: 081_pas_chartist2.txt
24423: 082_pas_biorhythmus.txt
24424: 082_pas_biorhythmus_solution.txt
24425: 082_pas_biorhythmus_solution_3.txt
24426: 082_pas_biorhythmus_test.txt
24427: 083_pas_GITARRE.txt
24428: 083_pas_soundbox_tones.txt
24429: 084_pas_waves.txt
24430: 085_mxsinus_logo.txt
24431: 085_sinus_plot_waves.txt
24432: 086_pas_graph_arrow_heart.txt
24433: 087_bitmap_loader.txt
24434: 087_pas_bitmap_solution.txt
24435: 087_pas_bitmap_solution2.txt
24436: 087_pas_bitmap_subimage.txt
24437: 087_pas_bitmap_test.txt
24438: 088_pas_soundbox2_mp3.txt
24439: 088_pas_soundbox_mp3.txt

334_jvutils_u.txt
335_atomimage5.txt
335_atomimage6.txt
335_atomimage7.txt
336_digiclock.txt
336_digiclock2.txt
336_digiclock2test.txt
336_digiclock3.txt
337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt
339_ntfs.txt
340_docctype.txt
340_logsimulation.txt
340_logsimulation2.txt
340_soundControltype.txt
341_blix_clock.txt
341_blix_clock2.txt
341_blix_clock_tester.txt
342_set_enumerator.txt
343_dice2.txt
344_pe_header.txt
344_pe_header2.txt
345_velocity.txt
346_conversions.txt
347_pictureview.txt
348_duallistview.txt
349_binteger.txt
350_parserform.txt
351_chartform.txt
351_chartform2.txt
351_chartform3.txt
352_array_unittest.txt
353_smtp_email.txt
353_smtp_email2.txt
354_josephus.txt
355_life_of_PI.txt
356_3D_printer.txt
357_fplot.TXT
358_makesound.txt
359_charsetrules.TXT
360_allobjects.TXT
360_JvPaintFX.TXT
361_heartbeat_wave.TXT
362_maxonmotor2.TXT
363_compress_services.txt
363_compress_services2.txt
364_pdf_services.txt
365_memorystream.txt
365_memorystream2.txt
365_memorystream_test.txt
365_U_HexView.txt
366_mp3player.txt
366_mp3player2.txt
366_mp3player2_themestest.txt
367_silvi_player_widgets.txt
367_silvi_player_widgets2.txt
367_widgets.txt
368_configuration_demo.txt
369_macro_demo.txt
370_callback2grid.TXT
370_richedit.txt
370_richedit_highlight.txt
370_synedit.txt
370_synedit2.txt
370_synedit2_mxtester.txt
370_synedit2_mxtester2.txt
371_maxbook_v4tester.txt
372_stackibz2_memoryalloc.TXT
372_synedit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt
374_realtime_randomtest.txt
374_realtime_randomtest2.txt
375_G9_musicbox.txt
376_collections_list.txt
377_simpleXML.txt
377_smartXML.txt
377_smartXMLWorkshop.txt
377_smartXMLWorkshop2.txt
378_queryperformance3.txt
378_REST1.txt
378_REST2.txt
379_timefunc.txt
379_timefuncTesterfilemon.txt
380_coolfunc.txt

```

24440: 088_pas_sphere_2.txt	380_coolfunc2.txt
24441: 089_pas_gradient.txt	380_coolfunc_tester.txt
24442: 089_pas_maxland2.txt	381_bitcoin_simulation.txt
24443: 090_pas_sudoku4.txt	382_GRMATH.TXT
24444: 090_pas_sudoku4_2.txt	382_GRMATH_PI_Proof.TXT
24445: 091_pas_cube4.txt	382_GRMATH_Riemann.TXT
24446: 092_pas_statistics4.txt	383_MDAC_DCOM.txt
24447: 093_variance.txt	384_TeamViewerID.TXT
24448: 093_variance_debug.txt	386_InternetRadio.TXT
24449: 094_pas_daysold.txt	387_fulltextfinder.txt
24450: 094_pas_stat_date.txt	387_fulltextfinder_cleancode.txt
24451: 095_pas_ki_simulation.txt	387_fulltextfinder_fast.txt
24452: 096_pas_geisen_problem.txt	387_fulltext_getscripttest.txt
24453: 096_pas_montyhall_problem.txt	388_TCPServerSock.TXT
24454: 097_lotto_proofofconcept.txt	388_TCPServerSock2.TXT
24455: 097_pas_lottocombinations_beat_plus.txt	388_TCPServerSockClient.TXT
24456: 097_pas_lottocombinations_beat_plus2.txt	389_TAR_Archive.TXT
24457: 097_pas_lottocombinations_universal.txt	389_TAR_Archive_test.TXT
24458: 097_pas_lottosimulation.txt	389_TAR_Archive_test2.TXT
24459: 098_pas_chartgenerator_plus.txt	390_Callback3.TXT
24460: 099_pas_3D_show.txt	390_Callback3Rec.TXT
24461: 200_big_numbers.txt	390_CallbackClean.TXT
24462: 200_big_numbers2.txt	390_StringlistHTML.TXT
24463: 201_streamload_xml.txt	391_ToDo_List.TXT
24464: 202_systemcheck.txt	392_Barcode.TXT
24465: 203_webservice_simple_intftester.txt	392_Barcode2.TXT
24466: 204_webservice_simple.txt	392_Barcode23.TXT
24467: 205_future_value_service.txt	392_Barcode2scholz.TXT
24468: 206_DTD_string_functions.txt	392_Barcode3scholz.TXT
24469: 207_ibz2_async_process.txt	393_QRCode.TXT
24470: 208_crc32_hash.txt	393_QRCode2.TXT
24471: 209_cryptohash.txt	393_QRCode2Direct.TXT
24472: 210_public_private.txt	393_QRCode2DirectIndy.TXT
24473: 210_public_private_cryptosystem.txt	393_QRCode2Direct_detlef.TXT
24474: 211_wipe_pattern.txt	393_QRCode3.TXT
24475: 211_wipe_pattern2.txt	394_networkgraph.TXT
24476: 211_wipe_pattern_solution.txt	394_networkgraph_depwalkutilstest.TXT
24477: 212_pas_statisticmodule4.TXT	394_networkgraph_depwalkutilstest2.TXT
24478: 212_pas_statisticmodulext.TXT	395_USBController.TXT
24479: 212_statisticmodule4.txt	396_Sort.TXT
24480: 213_pas_BBP_Algo.txt	397_Hotlog.TXT
24481: 214_mxdocudemo.txt	397_Hotlog2.TXT
24482: 214_mxdocudemo2.txt	398_ustrings.txt
24483: 214_mxdocudemo3.txt	399_form_templates.txt
24484: 215_hints_test.TXT	400_fploottchart.TXT
24485: 216_warnings_test.TXT	400_fploottchart2.TXT
24486: 217_pas_heartbeat.txt	400_fploottchart2teetest.TXT
24487: 218_biorhythmus_studio.txt	400_QRCodeMarket.TXT
24488: 219_cipherbox.txt	401_tfilerun.txt
24489: 219_crypt_source_comtest_solution.TXT	402_richedit2.txt
24490: 220_cipherbox_form.txt	403_outlookspy.txt
24491: 220_cipherbox_form2.txt	404_simplebrowser.txt
24492: 221_bcd_explain.txt	405_datefinder_today.txt
24493: 222_memoform.txt	406_portscan.txt
24494: 223_directorybox.txt	407_indydemo.txt
24495: 224_dialogs.txt	408_testroboter.txt
24496: 225_sprite_animation.txt	409_excel_control.txt
24497: 226_ASCII_Grid2.TXT	410_keyboardevent.txt
24498: 227_animation.txt	411_json_test.txt
24499: 227_animation2.txt	412_Zeosutils.txt
24500: 228_android_calendar.txt	413_listview2.txt
24501: 229_android_game.txt	414_avrdude_flash.txt
24502: 229_android_game_tester.txt	415_avrdude_writehex.txt
24503: 230_DataProvider.txt	416_sonar_startscriptEKON.TXT
24504: 230_DataSetProvider.txt	416_sonar_startscriptEKON_reporting.TXT
24505: 230_DataSetXMLBackupScholz.txt	417_GRMATH_PI_Proof2.TXT
24506: 231_DBGrid_access.txt	418_functional_paradigm.txt
24507: 231_DBGrid_XMLaccess.txt	419_archimedes_spiral.txt
24508: 231_DBGrid_XMLaccess2.txt	419_archimedes_spiral2.txt
24509: 231_DBGrid_XMLaccess_locatetester.txt	420_archimedes_arduino.txt
24510: 231_DBGrid_XML_CDS_local.txt	420_Lissajous.txt
24511: 232_outline.txt	421_PI_Power.TXT
24512: 232_outline_2.txt	421_PI_Power2.TXT
24513: 233_modular_form.txt	422_world_bitboxx.txt
24514: 234_debugoutform.txt	423_game_of_life.TXT
24515: 235_fastform.TXT	423_game_of_life2.TXT
24516: 236_componentpower.txt	423_game_of_life3.TXT
24517: 236_componentpower_back.txt	423_game_of_life3_test.TXT
24518: 237_pas_4forms.txt	423_game_of_life4.TXT
24519: 238_lottogen_form.txt	423_game_of_life4_kryptum.TXT
24520: 239_pas_sierpinski.txt	424_opengl_tester.txt
24521: 239_pas_sierpinski2.txt	425_reversi_game.txt
24522: 240_unitGlobal_tester.txt	426_IBUtils.TXT
24523: 241_db3_sql_tutorial.txt	427_IBDatabase.TXT
24524: 241_db3_sql_tutorial2.txt	428_SortGrid.TXT
24525: 241_db3_sql_tutorial2fix.txt	429_fileclass.txt
24526: 241_db3_sql_tutorial3.txt	430_fileoperation.txt
24527: 241_db3_sql_tutorial3connect.txt	430_fileoperation_tester.txt
24528: 241_db3_sql_tutorial3_fpptest.txt	431_performance_index.txt

```

24529: 241_RTL_SET2.txt
24530: 241_RTL_SET2_tester.txt
24531: 242_Component_Control.txt
24532: 243_tutorial_loader.txt
24533: 244_script_loader_loop.txt
24534: 245_formapp2.txt
24535: 245_formapp2_tester.txt
24536: 245_formapp2_testerX.txt
24537: 246_httpapp.txt
24538: 247_datecalendar.txt
24539: 248_ASCII_Grid2_sorted.TXT
24540: 249_picture_grid.TXT
24541: 250_tipsandtricks2.txt
24542: 250_tipsandtricks3.txt
24543: 250_tipsandtricks3api.txt
24544: 250_tipsandtricks3_admin_elevation.txt
24545: 250_tipsandtricks3_tester.txt
24546: 250_tipsandtricks4_tester.txt
24547: 250_tipsandtricks4_tester2.txt
24548: 251_compare_noise_gauss.txt
24549: 251_whitenoise.txt
24550: 251_whitenoise2.txt
24551: 252_hilbert_turtle.txt
24552: 252_pas_hilbert.txt
24553: 253_opearatingsystem3.txt
24554: 254_dynarrays.txt
24555: 255_einstein.txt
24556: 256_findconsts_of_EXE.txt
24557: 256_findfunctions2_of_EXE.txt
24558: 256_findfunctions2_of_EXEaverp.txt
24559: 256_findfunctions2_of_EXEspec.txt
24560: 256_findfunctions3.txt
24561: 256_findfunctions_of_EXE.txt
24562: 257_AES_Cipher.txt
24563: 258_AES_cryptobox.txt
24564: 258_AES_cryptobox2.txt
24565: 258_AES_cryptobox2_passdlg.txt
24566: 259_AES_crypt_directory.txt
24567: 260_sendmessage_2.TXT
24568: 260_sendmessage_beta.TXT
24569: 261_probability.txt
24570: 262_mxoutputdemo4.txt
24571: 263_async_sound.txt
24572: 264_vclutils.txt
24573: 264_VCL_utils2.txt
24574: 265_timer_API.txt
24575: 266_serial_interface.txt
24576: 266_serial_interface2.txt
24577: 266_serial_interface3.txt
24578: 267_ackermann_rec.txt
24579: 267_ackermann_variants.txt
24580: 268_DBGrid_tree.txt
24581: 269_record_grid.TXT
24582: 270_Jedi_FunctionPower.txt
24583: 270_Jedi_FunctionPowertester.txt
24584: 271_closures_study.txt
24585: 271_closures_study_workingset2.txt
24586: 272_pas_function_show.txt
24587: 273_pas_function_show2.txt
24588: 274_library_functions.txt
24589: 275_turtle_language.txt
24590: 275_turtle_language_save.txt
24591: 276_save_algo.txt
24592: 276_save_algo2.txt
24593: 277_functionsfor39.txt
24594: 278_DB_Dialogs.TXT
24595: 279_hexer2.TXT
24596: 279_hexer2macro.TXT
24597: 279_hexer2macroback.TXT
24598: 280_UML_process.txt
24599: 280_UML_process_knabe2.txt
24600: 280_UML_process_knabe3.txt
24601: 280_UML_process_TIM_Botzenhardt.txt
24602: 280_UML_TIM_Seitz.txt
24603: 281_picturepuzzle.txt
24604: 281_picturepuzzle2.txt
24605: 281_picturepuzzle3.txt
24606: 281_picturepuzzle4.txt
24607: 479_inputquery.txt
24608: 480_regex_pathfinder2.txt
24609: 482_processPipe.txt
24610: 483_PathFuncTest_mX.txt
24611: 485_InnoFunc.txt
24612: 487_asyncKeyState.txt
24613: 489_simpleComport.txt
24614: 491_analogmeter.txt
24615: 493_gadgets.txt
24616: 496_InstallX.txt
24617: 498_UnitTesting.txt

432_shortstring_routines.txt
433_video_avicap.txt
433_video_avicap2.txt
434_GSM_module.TXT
435_httpcommon.txt
436_GraphicSplitter.txt
436_GraphicSplitter_form.txt
436_GraphicSplitter_form2.txt
436_teetest_screen.TXT
436_teetest_screen2.TXT
437_WinAPItop.txt
437_WinAPItop_Firebirdtester.txt
438_OvcInternational.txt
439_AsyncFreeDemo.txt
439_AsyncFreeDemoForm.txt
440_DLL_Tutor.txt
440_DLL_Tutor2.txt
440_XML_Tutor.txt
440_XML_Tutor2.txt
441_make_app.txt
442_arduino_rgb_led.txt
443_webserver_arduino_rgb_light.txt
443_webserver_arduino_rgb_light4.txt
444_webserver_arduino3ibz_rgb_led_basta.txt
445_datagrid.txt
445_datagrid2.txt
445_datagrid_android_arduino.txt
446_arduino_timer.txt
447_patternFrm_mx3.txt
448_Synapse.txt
448_Synapse2.txt
449_dweb_start_tester.txt
450_Synapse_HTTPS.txt
450_Synapse_Mime.txt
450_Synapse_ScanPing.txt
451_ocx_player.txt
451_OCX_WinPlayer2.txt
452_dbtreeview.txt
452_dbtreeview2.txt
453_stdfuncs.txt
454_fileStream.txt
455_functionfun.txt
455_functionfun2.txt
455_functionfun2_test.txt
457_ressource_grid.txt
458_atomimageX.txt
459_cindyfunc.txt
459_cindyfunc2.txt
460_TopTenFunctions.txt
461_sqlform_calvin.txt
462_caesarcipher.txt
463_global_exception.txt
464_function_procedure.txt
464_function_procedure2.txt
464_function_procedure3.txt
465_U_HexView.txt
466_moon.txt
466_moon_inputquery.txt
4671_cardmagic.txt
467_helmholtz_graphic.txt
468_URLMon.txt
468_URLMon2.txt
469_formarrow.txt
469_formarrow_datepicker.txt
469_formarrow_datepicker_ibz_result.txt
469_ibzresult.txt
470_DFUUtils_compiled.txt
470_DFUUtils_ScrollingLED.txt
470_Oscilloscope.txt
470_Oscilloscope_code.txt
471_cardmagic.txt
471_cardmagic2.txt
472_allcards.TXT
473_comboiset.txt
474_wakeonlan.txt
474_wakeonlan2.txt
476_getscripttest.txt
477_filenameonly.txt
480_regex_pathfinder.txt
481_processList.txt
482_processPipeGCC.txt
484_filefinder3.txt
486_VideoGrabber.txt
488_asyncTerminal.txt
490_webCamproc.txt
492_snowflake2.txt
495_fourierfreq.txt
497_LED.txt
499_mulu42.txt

```

```

24618: 500_diceoflives.txt          501_firebird_datasnap_tests.txt
24619: 502_findalldocs.txt        503_led_switch.txt
24620: 504_fileclass.txt          505_debug.txt
24621: 506_colormatrix.txt        507_derutils.txt
24622: 508_simplecomportmorse.txt 509_GEOMap2.txt
24623: 509_509_GEOMap2_SReverse.TXT 510_510_bonn_gpsdata_mx4.pas
24624: 511_LEDLabel.txt          512_LED_moon.txt
24625: 513_StreamIntegration.txt 514_LED_moon2.txt
24626: 515_ledclock3.txt          516_mapview.txt
24627: 517_animation7.txt        518_sensors_meter.txt
24628: 519_powtils.txt           520_run_bytocode.txt
24629: 521_iputils2.txt          522_getgeocode.txt
24630: 523_NMEA.txt              524_NAV_Utils.txt
24631: 525_GEO84s.txt            526_Compass_meter.txt
24632: 527_GPSDemo.txt          528_linescount.txt
24633: 529_profilertest.txt      530_3DLab.txt
24634: 531_profilertest.txt      532_mcicommand.txt
24635: 533_syncasync_demo.txt    534_arduino_cockpit.TXT
24636: 535_Battleship3.pas       536_ressource_grid2.txt
24637: 537_iniplus.TXT          538_shellbatch.txt
24638: 539_timeturtle123.txt     540_NeuralNetwork.pas
24639: 541_webserver_arduino_motorturtle.txt 542_arduino_sound.txt
24640: 543_MATH_TurboP.PAS       544_UTIL01.PAS
24641: 545_strips.TXT           546_fourier3.pas
24642: 547_regexmaster.TXT       548_STExpressions.TXT -Services
24643: 549_3D_Panorama.txt       550_Expressions.TXT - 550_ADO_OLEDB.txt
24644: 551_ArduinoTester.txt     552_WaitExec32.txt
24645: 553_ArduinoCockBit3.txt   554_Watcdchdog.txt - 555_CODEsign2.txt
24646: 556_stringlistrandom.TXT  557_4dice2015.txt
24647: 558_highrestimer.TXT     -559_highthrestimer2
24648: 561_newfunctions399160.txt 560_PSUtils.TXT
24649: 563_moonpaper.txt         562_shellctrldemo.txt
24650: 565_ConsoleCapture.txt    564_queryperformance.txt
24651: 567_SquareWordGrids2.txt  566_queryperformance2.txt
24652: 569_keylog.txt - 569_ServiceMgr2.TXT 568_U_BigFloatTestscript2.pas
24653: 571_myPing.txt           570_turingsspeech.txt
24654: 573_modbusfrm_Main.pas    572_shellctrlplus.txt
24655: 575_TARTARUGA/Desktop.txt 574_arduino_cockpit5.TXT
24656: 577_listbox2list.txt      576_outlineEX11.PAS
24657: 579_numbersystems_sort.txt 578_access_db_logsimulation3.txt
24658: 581_stringstream.txt      580_indystacksearch_geo.txt
24659: 583_VirtualConstructor_savereport.txt 582_indystackwin.txt
24660: 585_fulltextfinder_cleancode_override.txt 584_ProcessList2fontwidth.txt last
24661: 587_one_function.txt      586_SRandom.txt
24662: 589_avi_animate.txt       588_XSBuiltins.txt
24663: 591_emailattach.txt       590_HTML_to_RTF.txt
24664: 593_round_time.txt        592_getTypeLibList.txt
24665: 595_check_memory.txt     594_check_creditcard.txt
24666: 597_ole_commands.txt     596_time_delays.txt
24667: 599_bug.txt              598_software_list.txt
24668: 601_PECheckSum.txt       600_surprise_nice.txt
24669: 603_cupids_arrow.TXT     602_multilang_game.txt (moonbug)
24670: 605_maxonmotor3DTage2BASTA2015.TXT 604_GEOCodeReverse4.TXT
24671: 607_DataSetProvider_CDS_ADO.txt 606_U_FibonacciSunflower.TXT
24672: 609_ScriptExecutor (beta) 608_ColorMixer_Arduino2.txt
24673: 611_Arduino_COMOutputs.txt 610_3D_DLL.txt
24674: 613_uPSI_DIUtils_test.pas 612_MTerminal.pas
24675: 615 SONAR_51_Starter.txt  614_inbrowserock.txt
24676:                           616_ComTerminalDlg.pas
24677: (589_AVI_Throbber.res.txt) 589_avi_animate.txt
24678:
24679: http://sourceforge.net/projects/maxbox/files/Examples/
24680:
24681: Help Online:
24682: http://max.kleiner.com/maxbox_functions_all.htm
24683:
24684: WebScript Examples:
24685:
24686: http://www.softwareschule.ch/examples/performer.txt;
24687: http://www.softwareschule.ch/examples/turtle.txt;
24688: http://www.softwareschule.ch/examples/SQLExport.txt;
24689: http://www.softwareschule.ch/examples/Richter.txt;
24690: http://www.softwareschule.ch/examples/checker.txt;
24691: http://www.softwareschule.ch/examples/demoscript.txt;
24692: http://www.softwareschule.ch/examples/ibzresult.txt;
24693: http://www.softwareschule.ch/examples/performindex.txt
24694: http://www.softwareschule.ch/examples/processlist.txt
24695: http://www.softwareschule.ch/examples/game.txt
24696: http://www.softwareschule.ch/examples/GEOGPS.txt
24697: http://www.softwareschule.ch/examples/turtle2.txt
24698: http://www.softwareschule.ch/examples/turtle3.txt
24699: http://www.softwareschule.ch/examples/asyncterminal.txt
24700: http://www.softwareschule.ch/examples/snowflake.txt
24701: http://www.softwareschule.ch/examples/arduinoled.txt
24702: http://www.softwareschule.ch/examples/moon2.txt
24703: http://www.softwareschule.ch/examples/cockpit.txt
24704: http://www.softwareschule.ch/examples/tartaruga.txt
24705: http://www.softwareschule.ch/examples/surprise.txt
24706:

```

```

24707:
24708: Delphi Basics Run Time Library listing
24709: ****
24710: A
24711: Compiler Directive $A Determines whether data is aligned or packed
24712: Compiler Directive $Align Determines whether data is aligned or packed
24713: Compiler Directive $AppType Determines the application type : GUI or Console
24714: Procedure SysUtils Abort Aborts the current processing with a silent exception
24715: Function System Abs Gives the absolute value of a number (-ve sign is removed)
24716: Directive Abstract Defines a class method only implemented in subclasses
24717: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
24718: Function System Addr Gives the address of a variable, function or procedure
24719: Keyword And Boolean and or bitwise and of two arguments
24720: Type System AnsiChar A character type guaranteed to be 8 bits in size
24721: Function SysUtils AnsiCompareStr Compare two strings for equality
24722: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
24723: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
24724: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
24725: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
24726: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
24727: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
24728: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
24729: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string
24730: Function StrUtils AnsiPos Find the position of one string in another
24731: Function StrUtils AnsiReplaceStr Replaces a part of one string with another
24732: Function StrUtils AnsiReverseString Reverses the sequence of letters in a string
24733: Function StrUtils AnsiRightStr Extracts characters from the right of a string
24734: Function StrUtils AnsiStartsStr Returns true if a string starts with a substring
24735: Type System AnsiString A data type that holds a string of AnsiChars
24736: Function SysUtils AnsiUpperCase Change lower case characters in a string to upper case
24737: Procedure System Append Open a text file to allow appending of text to the end
24738: Procedure SysUtils AppendStr Concatenate one string onto the end of another
24739: Function Math ArcCos The Arc Cosine of a number, returned in radians
24740: Function Math ArcSin The Arc Sine of a number, returned in radians
24741: Function System ArcTan The Arc Tangent of a number, returned in radians
24742: Keyword Array A data type holding indexable collections of data
24743: Keyword As Used for casting object references
24744: Procedure System Assign Assigns a file handle to a binary or text file
24745: Function System Assigned Returns true if a reference is not nil
24746: Procedure System AssignFile Assigns a file handle to a binary or text file
24747: Procedure Printers AssignPrn Treats the printer as a text file - an easy way of printing text
24748:
24749: B
24750: Compiler Directive $B Whether to short cut and and or operations
24751: Compiler Directive $BoolEval Whether to short cut and and or operations
24752: Procedure SysUtils Beep Make a beep sound
24753: Keyword Begin Keyword that starts a statement block
24754: Function System BeginThread Begins a separate thread of code execution
24755: Procedure System BlockRead Reads a block of data records from an untyped binary file
24756: Procedure System BlockWrite Writes a block of data records to an untyped binary file
24757: Type System Boolean Allows just True and False values
24758: Function Classes Bounds Create a TRect value from top left and size values
24759: Procedure System Break Forces a jump out of a single loop
24760: Type System Byte An integer type supporting values 0 to 255
24761:
24762: C
24763: Type System Cardinal The basic unsigned integer type
24764: Keyword Case A mechanism for acting upon different values of an Ordinal
24765: Function StdConv CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
24766: Function SysUtils ChangeFileExt Change the extension part of a file name
24767: Type System Char Variable type holding a single character
24768: Procedure System ChDir Change the working drive plus path for a specified drive
24769: Function System Chr Convert an integer into a character
24770: Keyword Class Starts the declaration of a type of object class
24771: Procedure System Close Closes an open file
24772: Procedure System CloseFile Closes an open file
24773: Variable System CmdLine Holds the execution text used to start the current program
24774: Type System Comp A 64 bit signed integer
24775: Function SysUtils CompareStr Compare two strings to see which is greater than the other
24776: Function SysUtils CompareText Compare two strings for equality, ignoring case
24777: Function Math CompareValue Compare numeric values with a tolerance
24778: Function System Concat Concatenates one or more strings into one string
24779: Keyword Const Starts the definition of fixed data values
24780: Keyword Constructor Defines the method used to create an object from a class
24781: Procedure System Continue Forces a jump to the next iteration of a loop
24782: Function ConvUtils Convert Convert one measurement value to another
24783: Function System Copy Create a copy of part of a string or an array
24784: Function System Cos The Cosine of a number
24785: Function SysUtils CreateDir Create a directory
24786: Type System Currency A floating point type with 4 decimals used for financial values
24787: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
24788: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
24789: Variable SysUtils CurrencyString The currency string used in currency display functions
24790: Function SysUtils CurrToStr Convert a currency value to a string
24791: Function SysUtils CurrToStrF Convert a currency value to a string with formatting
24792:
24793: D
24794: Compiler Directive $D Determines whether application debug information is built
24795: Compiler Directive $DebugInfo Determines whether application debug information is built

```

24796: Compiler Directive **\$Define** Defines a compiler directive symbol - **as** used by IfDef
 24797: Compiler Directive **\$DefinitionInfo** Determines whether application symbol information is built
 24798: **Function** SysUtils Date Gives the current date
 24799: Variable SysUtils DateSeparator The character used **to** separate display date fields
 24800: **Function** SysUtils DateTimeToFileDate Convert a TDateTime value **to** a **File** date/time format
 24801: **Function** SysUtils DateTimeToStr Converts TDateTime date **and** time values **to** a **string**
 24802: **Procedure** SysUtils DateTimeToString Rich formatting **of** a TDateTime variable into a **string**
 24803: **Function** SysUtils DateToStr Converts a TDateTime date value **to** a **string**
 24804: **Function** DateUtils DayOfTheMonth Gives day **of** month index **for** a TDateTime value (ISO 8601)
 24805: **Function** DateUtils DayOfTheWeek Gives day **of** week index **for** a TDateTime value (ISO 8601)
 24806: **Function** DateUtils DayOfTheYear Gives the day **of** the year **for** a TDateTime value (ISO 8601)
 24807: **Function** SysUtils DayOfWeek Gives day **of** week index **for** a TDateTime value
 24808: **Function** DateUtils DaysBetween Gives the whole number **of** days between **2** dates
 24809: **Function** DateUtils DaysInAMonth Gives the number **of** days **in** a month
 24810: **Function** DateUtils DaysInAYear Gives the number **of** days **in** a year
 24811: **Function** DateUtils DaySpan Gives the fractional number **of** days between **2** dates
 24812: **Procedure** System Dec Decrement an ordinal variable
 24813: Variable SysUtils DecimalSeparator The character used **to** display the decimal point
 24814: **Procedure** SysUtils DecodeDate Extracts the year, month, day values from a TDateTime **var**.
 24815: **Procedure** DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
 24816: **Procedure** SysUtils DecodeTime Break a TDateTime value into individual time values
 24817: Directive **Default** Defines default processing **for** a **property**
 24818: **Function** Math DegToRad Convert a degrees value **to** radians
 24819: **Procedure** System Delete Delete a section **of** characters from a **string**
 24820: **Function** SysUtils Deletefile Delete a **file** specified by its **file** name
 24821: Keyword **Destructor** Defines the method used **to** destroy an **object**
 24822: **Function** SysUtils DirectoryExists Returns true **if** the given directory exists
 24823: **Function** SysUtils DiskFree Gives the number **of** free bytes **on** a specified drive
 24824: **Function** SysUtils DiskSize Gives the size **in** bytes **of** a specified drive
 24825: **Procedure** System Dispose Dispose **of** storage used by a pointer **type** variable
 24826: Keyword **Div** Performs integer division, discarding the remainder
 24827: Keyword **Do** Defines the start **of** some controlled action
 24828: **Type** System Double A floating point **type** supporting about **15** digits **of** precision
 24829: Keyword **DownTo** Prefixes an decremental **for** loop target value
 24830: **Function** StrUtils DupeString Creates a **string** containing copies **of** a substring
 24831: Directive **Dynamic** Allows a **class** method **to** be overriden **in** derived classes
 24832:
 24833: E
 24834: Compiler Directive **\$Else** Starts the alternate section **of** an IfDef **or** IfNDef
 24835: Compiler Directive **\$EndIf** Terminates conditional code compilation
 24836: Compiler Directive **\$ExtendedSyntax** Controls some **Pascal** extension handling
 24837: Keyword **Else** Starts false section **of** if, case **and** try statements
 24838: **Function** SysUtils EncodeDate Build a TDateTime value from year, month **and** day values
 24839: **Function** DateUtils EncodeDateTime Build a TDateTime value from day **and** time values
 24840: **Function** SysUtils EncodeTime Build a TDateTime value from hour, min, sec **and** msec values
 24841: Keyword **End** Keyword that terminates statement blocks
 24842: **Function** DateUtils EndOfDay Generate a TDateTime value **set** to the very **end** **of** a day
 24843: **Function** DateUtils EndOfMonth Generate a TDateTime value **set** to the very **end** **of** a month
 24844: **Procedure** System EndThread Terminates a thread **with** an exit code
 24845: **Function** System EOF Returns true **if** a **file** opened **with** Reset **is** at the **end**
 24846: **Function** System Eoln Returns true **if** the current text **file** **is** pointing at a line **end**
 24847: **Procedure** System Erase Erase a **file**
 24848: Variable System ErrorAddr Sets the error address when an application terminates
 24849: Keyword **Except** Starts the error trapping clause **of** a Try statement
 24850: **Procedure** System Exclude Exclude a value **in** a set variable
 24851: **Procedure** System Exit Exit abruptly from a **function** **or** **procedure**
 24852: Variable System ExitCode Sets the return code when an application terminates
 24853: **Function** System Exp Gives the exponent **of** a number
 24854: Directive System **Export** Makes a **function** **or** **procedure** **in** a DLL externally available
 24855: **Type** System Extended The floating point **type** with the highest capacity **and** precision
 24856: **Function** SysUtils ExtractFileDir Extracts the dir part **of** a full **file** name
 24857: **Function** SysUtils ExtractFileDrive Extracts the drive part **of** a full **file** name
 24858: **Function** SysUtils ExtractFileExt Extracts the extension part **of** a full **file** name
 24859: **Function** SysUtils ExtractFileName Extracts the name part **of** a full **file** name
 24860: **Function** SysUtils ExtractFilePath Extracts the path part **of** a full **file** name
 24861:
 24862: F
 24863: **Function** StdConv FahrenheitToCelsius Convert a fahrenheit temperature into celsius
 24864: Keyword **File** Defines a typed **or** untyped **file**
 24865: **Function** SysUtils FileAge Get the last modified date/time **of** a **file** without opening it
 24866: **Function** SysUtils FileDateToDateTime Converts a **file** date/time format **to** a TDateTime value
 24867: **Function** SysUtils FileExists Returns true **if** the given **file** exists
 24868: **Function** SysUtils FileGetAttr Gets the attributes **of** a **file**
 24869: Variable System FileMode Defines how Reset opens a binary **file**
 24870: **Function** System FilePos Gives the **file** position **in** a binary **or** text **file**
 24871: **Function** SysUtils FileSearch Search **for** a **file** **in** one **or** more directories
 24872: **Function** SysUtils FileSetAttr Sets the attributes **of** a **file**
 24873: **Function** SysUtils FileSetDate Set the last modified date **and** time **of** a **file**
 24874: **Function** System FileSize Gives the size **in** records **of** an open **file**
 24875: **Procedure** System FillChar Fills **out** a section **of** storage **with** a fill character **or** byte value
 24876: Keyword **Finally** Starts the unconditional code section **of** a Try statement
 24877: **Function** SysUtils FindClose Closes a successful FindFirst **file** search
 24878: **Function** SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
 24879: **Function** SysUtils FindFirst Finds all files matching a **file** mask **and** attributes
 24880: **Function** SysUtils FindNext Find the next **file** after a successful FindFirst
 24881: **Function** SysUtils FloatToStr Convert a floating point value **to** a **string**
 24882: **Function** SysUtils FloatToStrF Convert a floating point value **to** a **string** **with** formatting
 24883: **Procedure** System Flush Flushes buffered text **file** data **to** the **file**
 24884: Keyword **For** Starts a loop that executes a finite number **of** times

```

24885: Function SysUtils ForceDirectories Create a new path of directories
24886: Function SysUtils Format Rich formatting of numbers and text into a string
24887: Function SysUtils FormatCurr Rich formatting of a currency value into a string
24888: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
24889: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
24890: Function System Frac The fractional part of a floating point number
24891: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil
24892: Procedure System FreeMem Free memory storage used by a variable
24893: Keyword System Function Defines a subroutine that returns a value
24894:
24895: G
24896: Function SysUtils GetCurrentDir Get the current directory (drive plus directory)
24897: Procedure System GetDir Get the default directory (drive plus path) for a specified drive
24898: Function System GetLastError Gives the error code of the last failing Windows API call
24899: Procedure SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
24900: Function System GetMem Get a specified number of storage bytes
24901: Keyword Goto Forces a jump to a label, regardless of nesting
24902:
24903: H
24904: Compiler Directive $H Treat string types as AnsiString or ShortString
24905: Compiler Directive $Hints Determines whether Delphi shows compilation hints
24906: Procedure System Halt Terminates the program with an optional dialog
24907: Function System Hi Returns the hi-order byte of a (2 byte) Integer
24908: Function System High Returns the highest value of a type or variable
24909:
24910: I
24911: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
24912: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
24913: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
24914: Compiler Directive $IfOpt Tests for the state of a Compiler directive
24915: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
24916: Compiler Directive $IOChecks When on, an IO operation error throws an exception
24917: Keyword If Starts a conditional expression to determine what to do next
24918: Keyword Implementation Starts the implementation (code) section of a Unit
24919: Keyword In Used to test if a value is a member of a set
24920: Procedure System Inc Increments an ordinal variable
24921: Function DateUtils IncDay Increments a TDateTime variable by + or - number of days
24922: Procedure System Include Include a value in a set variable
24923: Function DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
24924: Function DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
24925: Function SysUtils IncMonth Increments a TDateTime variable by a number of months
24926: Function DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
24927: Function DateUtils IncYear Increments a TDateTime variable by a number of years
24928: Directive Index Principally defines indexed class data properties
24929: Constant Math Infinity Floating point value of infinite size
24930: Keyword Inherited Used to call the parent class constructor or destructor method
24931: Variable System Input Defines the standard input text file
24932: Function Dialogs InputBox Display a dialog that asks for user text input, with default
24933: Function Dialogs InputQuery Display a dialog that asks for user text input
24934: Procedure System Insert Insert a string into another string
24935: Function System Int The integer part of a floating point number as a float
24936: Type System Int64 A 64 bit sized integer - the largest in Delphi
24937: Type System Integer The basic Integer type
24938: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
24939: Function SysUtils IntToHex Convert an Integer into a hexadecimal string
24940: Function SysUtils IntToStr Convert an integer into a string
24941: Function System IOResult Holds the return code of the last I/O operation
24942: Keyword Is Tests whether an object is a certain class or ascendant
24943: Function Math IsInfinite Checks whether a floating point number is infinite
24944: Function SysUtils IsLeapYear Returns true if a given calendar year is a leap year
24945: Function System IsMultiThread Returns true if the code is running multiple threads
24946: Function Math IsNaN Checks to see if a floating point number holds a real number
24947:
24948: L
24949: Compiler Directive $L Determines what application debug information is built
24950: Compiler Directive $LocalSymbols Determines what application debug information is built
24951: Compiler Directive $LongStrings Treat string types as AnsiString or ShortString
24952: Function SysUtils LastDelimiter Find the last position of selected characters in a string
24953: Function System Length Return the number of elements in an array or string
24954: Function System Ln Gives the natural logarithm of a number
24955: Function System Lo Returns the low-order byte of a (2 byte) Integer
24956: Function Math Log10 Gives the log to base 10 of a number
24957: Variable SysUtils LongDateFormat Long version of the date to string format
24958: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
24959: Type System LongInt An Integer whose size is guaranteed to be 32 bits
24960: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
24961: Variable SysUtils LongTimeFormat Long version of the time to string format
24962: Type System LongWord A 32 bit unsigned integer
24963: Function System Low Returns the lowest value of a type or variable
24964: Function SysUtils LowerCase Change upper case characters in a string to lower case
24965:
24966: M
24967: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
24968: Function Math Max Gives the maximum of two integer values
24969: Constant System MaxInt The maximum value an Integer can have
24970: Constant System MaxLongInt The maximum value an LongInt can have
24971: Function Math Mean Gives the average for a set of numbers
24972: Function Dialogs MessageDlg Displays a message, symbol, and selectable buttons
24973: Function Dialogs MessageDlgPos Displays a message plus buttons at a given screen position

```

24974: **Function** Math Min Gives the minimum **of** two integer values
 24975: Constant SysUtils MinsPerDay Gives the number **of** minutes **in** a day
 24976: **Procedure** System MkDir Make a directory
 24977: Keyword **Mod** Performs integer division, returning the remainder
 24978: Constant SysUtils MonthDays Gives the number **of** days **in** a month
 24979: **Function** DateUtils MonthOfTheYear Gives the month **of** the year **for** a TDateTime value
 24980: **Procedure** System Move Copy bytes **of** data from a source **to** a destination
 24981:
 24982: N
 24983: Constant Math NaN **Not** a real number
 24984: Variable SysUtils NegCurrFormat Defines negative amount formatting **in** currency displays
 24985: **Procedure** System New Create a new pointer **type** variable
 24986: Constant System **Nil** A pointer value that **is** defined **as** undetermined
 24987: Keyword **Not** Boolean **Not** or bitwise **not** **of** one arguments
 24988: **Function** SysUtils Now Gives the current date **and** time
 24989: Variable Variants Null A variable that has no value
 24990:
 24991: O
 24992: Compiler Directive \$O Determines whether Delphi optimises code when compiling
 24993: Compiler Directive \$Optimization Determines whether Delphi optimises code when compiling
 24994: Compiler Directive \$OverflowChecks Determines whether Delphi checks integer **and** enum bounds
 24995: Keyword System **Object** Allows a subroutine data **type** **to** refer **to** an **object** method
 24996: **Function** System Odd Tests whether an integer has an odd value
 24997: Keyword **Of** Linking keyword used **in** many places
 24998: Keyword **On** Defines exception handling **in** a **Try Except** clause
 24999: Keyword **Or** Boolean **or** or bitwise **or** **of** two arguments
 25000: **Function** System Ord Provides the Ordinal value **of** an integer, character **or** enum
 25001: Directive **Out** Identifies a routine parameter **for** output only
 25002: Variable System Output Defines the standard output text **file**
 25003: Directive **Overload** Allows **2** or more routines **to** have the same name
 25004: Directive **Override** Defines a method that replaces a **virtual** parent **class** method
 25005:
 25006: P
 25007: Keyword **Packed** Compacts complex data types into minimal storage
 25008: Type System PAnsiChar A pointer **to** an AnsiChar value
 25009: Type System PAnsiString Pointer **to** an AnsiString value
 25010: **Function** System ParamCount Gives the number **of** parameters passed **to** the current **program**
 25011: **Function** System ParamStr Returns one **of** the parameters used **to** run the current **program**
 25012: Type System PChar A pointer **to** an Char value
 25013: Type System PCurrency Pointer **to** a Currency value
 25014: Type System PDateTime Pointer **to** a TDateTime value
 25015: Type System PExtended Pointer **to** a Extended floating point value
 25016: **Function** System Pi The mathematical constant
 25017: Type System PInt64 Pointer **to** an Int64 value
 25018: **Function** Classes Point Generates a TPoint value from X **and** Y values
 25019: Type System Pointer Defines a general use Pointer **to** any memory based data
 25020: **Function** Classes PointsEqual Compares two TPoint values **for** equality
 25021: **Function** System Pos Find the position **of** one **string** **in** another
 25022: **Function** System Pred Decrement an ordinal variable
 25023: **Function** Printers Printer Returns a reference **to** the global Printer **object**
 25024: Directive **Private** Starts the section **of** private data **and** methods **in** a **class**
 25025: Keyword System **Procedure** Defines a subroutine that does **not** return a value
 25026: **Procedure** FileCtrl ProcessPath Split a drive/path/filename **string** into its constituent parts
 25027: Keyword System **Program** Defines the start **of** an application
 25028: **Function** Dialogs PromptForFileName Shows a dialog allowing the user **to** select a **file**
 25029: Keyword System **Property** Defines controlled access **to** **class** fields
 25030: Directive **Protected** Starts a section **of** class private data accesible **to** sub-classes
 25031: Type System PShortString A pointer **to** an ShortString value
 25032: Type System PString Pointer **to** a String value
 25033: **Function** Types PtInRect Tests **to** see **if** a point lies within a rectangle
 25034: Directive **Public** Starts an externally accessible section **of** a **class**
 25035: Directive **Published** Starts a published externally accessible section **of** a **class**
 25036: Type System PVariant Pointer **to** a Variant value
 25037: Type System PWideChar Pointer **to** a WideChar
 25038: Type System PWideString Pointer **to** a WideString value
 25039:
 25040: Q
 25041: Compiler Directive \$Q Determines whether Delphi checks integer **and** enum bounds
 25042:
 25043: R
 25044: Compiler Directive \$R Determines whether Delphi checks array bounds
 25045: Compiler Directive \$RangeChecks Determines whether Delphi checks array bounds
 25046: Compiler Directive \$ReferenceInfo Determines whether symbol reference information **is** built
 25047: Compiler Directive \$Resource Defines a resource **file** **to** be included **in** the application linking
 25048: **Function** Math RadToDeg Converts a radian value **to** degrees
 25049: Keyword **Raise** Raise an exception
 25050: **Function** System Random Generate a random floating point **or** integer number
 25051: **Procedure** System Randomize Reposition the Random number generator next value
 25052: **Function** Math RandomRange Generate a random integer number within a supplied range
 25053: Variable System RandSeed Reposition the Random number generator next value
 25054: **Procedure** System Read Read data from a binary **or** text **file**
 25055: **Procedure** System ReadLn Read a complete line **of** data from a text **file**
 25056: Type System Real A floating point **type** supporting about **15** digits **of** precision
 25057: Type System Real48 The floating point **type** **with** the highest capacity **and** precision
 25058: **Procedure** System ReallocMem Reallocate an existing block **of** storage
 25059: **Function** DateUtils RecodeDate Change only the date part **of** a TDateTime variable
 25060: **Function** DateUtils RecodeTime Change only the time part **of** a TDateTime variable
 25061: Keyword **Record** A structured data **type** - holding fields **of** data
 25062: **Function** Classes Rect Create a TRect value from **2** points **or** **4** coordinates

25063: **Function** SysUtils RemoveDir Remove a directory
 25064: **Procedure** System Rename Rename a **file**
 25065: **Function** SysUtils RenameFile Rename a **file** or directory
 25066: Keyword **Repeat** **Repeat** statements **until** a termination condition **is met**
 25067: **Procedure** SysUtils ReplaceDate Change only the date part **of** a TDateTime variable
 25068: **Procedure** SysUtils ReplaceTime Change only the time part **of** a TDateTime variable
 25069: **Procedure** System Reset Open a text **file** for reading, **or** binary **file** for **read/write**
 25070: Variable System Result A variable used **to** hold the return value from a **function**
 25071: **Procedure** System ReWrite Open a text **or** binary **file** for **write** access
 25072: **Procedure** System RmDir Remove a directory
 25073: **Function** System Round Rounds a floating point number **to** an integer
 25074: **Procedure** System RunError Terminates the **program** with an error dialog
 25075:
 25076: S
 25077: Constant SysUtils SecsPerDay Gives the number **of** seconds **in** a day
 25078: **Procedure** System Seek Move the pointer **in** a binary **file** **to** a new **record** position
 25079: **Function** System SeekEof Skip **to** the **end** **of** the current line **or** **file**
 25080: **Function** System SeekEoln Skip **to** the **end** **of** the current line **or** **file**
 25081: **Function** FileCtrl SelectDirectory Display a dialog **to** allow user selection **of** a directory
 25082: Variable System Self Hidden parameter **to** a method - refers **to** the containing **object**
 25083: Keyword **Set** Defines a **set** **of** up **to** 255 distinct values
 25084: **Function** SysUtils SetCurrentDir Change the current directory
 25085: **Procedure** System SetLength Changes the size **of** a **string**, **or** the size(s) **of** an **array**
 25086: **Procedure** System SetString Copies characters from a buffer **into** a **string**
 25087: Keyword **Shl** Shift an integer value left by a number **of** bits
 25088: Variable SysUtils ShortDateFormat Compact version **of** the date **to** **string** format
 25089: Variable SysUtils ShortDayNames An **array** **of** days **of** the week names, starting 1 = Sunday
 25090: **Type** System ShortInt An integer **type** supporting values -128 to 127
 25091: Variable SysUtils ShortMonthNames An **array** **of** days **of** the month names, starting 1 = Jan
 25092: **Type** System ShortString Defines a **string** **of** up **to** 255 characters
 25093: Variable SysUtils ShortTimeFormat Short version **of** the time **to** **string** format
 25094: **Procedure** Dialogs ShowMessage Display a **string** **in** a simple dialog **with** an OK button
 25095: **Procedure** Dialogs ShowMessageFmt Display formatted data **in** a simple dialog **with** an OK button
 25096: **Procedure** Dialogs ShowMessagePos Display a **string** **in** a simple dialog at a given screen position
 25097: Keyword **Shr** Shift an integer value right by a number **of** bits
 25098: **Function** System Sin The Sine **of** a number
 25099: **Type** System Single The smallest capacity **and** precision floating point **type**
 25100: **Function** System SizeOf Gives the storage byte size **of** a **type** **or** variable
 25101: **Function** System Slice Creates a slice **of** an **array** **as** an Open **Array** parameter
 25102: **Type** System SmallInt An Integer **type** supporting values from -32768 to 32767
 25103: **Function** System Sqr Gives the square **of** a number
 25104: **Function** System Sqrt Gives the square root **of** a number
 25105: **Procedure** System Str Converts an integer **or** floating point number **to** a **string**
 25106: **Type** System String A data **type** that holds a **string** **of** characters
 25107: **Function** System StringOfChar Creates a **string** **with** one character repeated many times
 25108: **Function** SysUtils StringReplace Replace one **or** more substrings found within a **string**
 25109: **Function** System StringToWideChar Converts a normal **string** **into** a WideChar 0 terminated buffer
 25110: **Function** SysUtils StrScan Searches **for** a specific character **in** a constant **string**
 25111: **Function** SysUtils StrToCurr Convert a number **string** **into** a currency value
 25112: **Function** SysUtils StrToDate Converts a date **string** **into** a TDateTime value
 25113: **Function** SysUtils StrToDateTime Converts a date+time **string** **into** a TDateTime value
 25114: **Function** SysUtils StrToFloat Convert a number **string** **into** a floating point value
 25115: **Function** SysUtils StrToInt Convert an integer **string** **into** an Integer value
 25116: **Function** SysUtils StrToInt64 Convert an integer **string** **into** an Int64 value
 25117: **Function** SysUtils StrToInt64Def Convert a **string** **into** an Int64 value **with** default
 25118: **Function** SysUtils StrToIntDef Convert a **string** **into** an Integer value **with** default
 25119: **Function** SysUtils StrToTime Converts a time **string** **into** a TDateTime value
 25120: **Function** StrUtils StuffString Replaces a part **of** one **string** **with** another
 25121: **Function** System Succ Increment an ordinal variable
 25122: **Function** Math Sum Return the sum **of** an **array** **of** floating point values
 25123:
 25124: T
 25125: **Function** Math Tan The Tangent **of** a number
 25126: **Type** Classes TBits An **object** that can hold an infinite number **of** Boolean values
 25127: Variable ConvUtils TConvFamily Defines a family **of** measurement types **as** used by Convert
 25128: **Type** ConvUtils TConvType Defines a measurement **type** **as** used by Convert
 25129: **Type** System TDateTime Data **type** holding a date **and** time value
 25130: **Type** System Text Defines a **file** **as** a text **file**
 25131: **Type** System TextFile Declares a **file** **type** **for** storing lines **of** text
 25132: **Type** SysUtils TFloatFormat Formats **for** use **in** floating point number display functions
 25133: **Type** SysUtils TFormatSettings A **record** **for** holding locale values **for** thread-safe functions
 25134: Keyword **Then** Part **of** an if statement - starts the true clause
 25135: Variable SysUtils ThousandSeparator The character used **to** display the thousands separator
 25136: Keyword **ThreadVar** Defines variables that are given separate instances per thread
 25137: **Function** SysUtils Time Gives the current time
 25138: Variable SysUtils TimeAMString Determines AM value **in** DateTimeToString **procedure**
 25139: Variable SysUtils TimePMString Determines PM value **in** DateTimeToString **procedure**
 25140: Variable SysUtils TimeSeparator The character used **to** separate display time fields
 25141: **Function** SysUtils TimeToStr Converts a TDateTime time value **to** a **string**
 25142: **Type** Classes TList General purpose container **of** a list **of** objects
 25143: Keyword **To** Prefixes an incremental **for** loop target value
 25144: **Type** System TObject The base **class** **type** **that** **is** ancestor **to** all other classes
 25145: **Function** DateUtils Tomorrow Gives the date tomorrow
 25146: **Type** Dialogs TOpenDialog Displays a **file** selection dialog
 25147: **Type** Types TPoint Holds X **and** Y integer values
 25148: **Type** Dialogs TPrintDialog **Class** **that** creates a printer selection **and** control dialog
 25149: **Type** Types TRect Holds rectangle coordinate values
 25150: **Type** SysUtils TReplaceFlags Defines options **for** the StringReplace routine
 25151: **Function** SysUtils Trim Removes leading **and** trailing blanks **from** a **string**

```

25152: Function SysUtils TrimLeft Removes leading blanks from a string
25153: Function SysUtils TrimRight Removes trailing blanks from a string
25154: Function System Trunc The integer part of a floating point number
25155: Procedure System Truncate Truncates a file size - removes all data after the current position
25156: Keyword Try Starts code that has error trapping
25157: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
25158: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
25159: Type Classes TStringList Holds a variable length list of strings
25160: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
25161: Type System TThreadFunc Defines the function to be called by BeginThread
25162: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
25163: Keyword Type Defines a new category of variable or process
25164:
25165: U
25166: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
25167: Keyword Unit Defines the start of a unit file - a Delphi module
25168: Keyword Until Ends a Repeat control loop
25169: Function System UpCase Convert a Char value to upper case
25170: Function SysUtils UpperCase Change lower case characters in a string to upper case
25171: Keyword Uses Declares a list of Units to be imported
25172:
25173: V
25174: Procedure System Val Converts number strings to integer and floating point values
25175: Keyword Var Starts the definition of a section of data variables
25176: Type System Variant A variable type that can hold changing data types
25177: Function Variants VarType Gives the current type of a Variant variable
25178: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
25179: Directive Virtual Allows a class method to be overriden in derived classes
25180:
25181: W
25182: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
25183: Keyword While Repeat statements whilst a continuation condition is met
25184: Type System WideChar Variable type holding a single International character
25185: Function System WideCharToString Copies a null terminated WideChar string to a normal string
25186: Type System WideString A data type that holds a string of WideChars
25187: Keyword With A means of simplifying references to structured variables
25188: Type System Word An integer type supporting values 0 to 65535
25189: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
25190: Procedure System Write Write data to a binary or text file
25191: Procedure System WriteLn Write a complete line of data to a text file
25192:
25193: X
25194: Compiler Directive $X Controls some Pascal extension handling
25195: Keyword Xor Boolean Xor or bitwise Xor of two arguments
25196: Y
25197: Compiler Directive $Y Determines whether application symbol information is built
25198: Function DateUtils Yesterday Gives the date yesterday
25199:
25200: Z
25201: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
25202:
25203:
25204:
25205: procedure SIRegister_uPSUtils(CL: TPPascalCompiler);
25206: begin
25207:   PSMainProcName', 'String').SetString( '!MAIN');
25208:   PSMainProcNameOrg', 'String').SetString( 'Main Proc');
25209:   'PSLowBuildSupport', 'LongInt').SetInt( 12);
25210:   CL.AddConstantN('PSCurrentBuildNo', 'LongInt').SetInt( 23);
25211:   'PSCurrentversion', 'String').SetString( '1.31');
25212:   'PSValidHeader', 'LongInt').SetInt( 1397769801);
25213:   'PSAddrStackStart', 'LongInt').SetInt( 1610612736);
25214:   'PSAddrNegativeStackStart', 'LongInt').SetInt( 1073741824);
25215:   'btReturnAddress', 'LongInt').SetInt( 0);
25216:   'btU8', 'LongInt').SetInt( 1);
25217:   'btS8', 'LongInt').SetInt( 2);
25218:   'btU16', 'LongInt').SetInt( 3);
25219:   'btS16', 'LongInt').SetInt( 4);
25220:   'btU32', 'LongInt').SetInt( 5);
25221:   'btS32', 'LongInt').SetInt( 6);
25222:   'btSingle', 'LongInt').SetInt( 7);
25223:   'btDouble', 'LongInt').SetInt( 8);
25224:   'btExtended', 'LongInt').SetInt( 9);
25225:   'btString', 'LongInt').SetInt( 10);
25226:   'btRecord', 'LongInt').SetInt( 11);
25227:   'btArray', 'LongInt').SetInt( 12);
25228:   'btPointer', 'LongInt').SetInt( 13);
25229:   'btPChar', 'LongInt').SetInt( 14);
25230:   'btResourcePointer', 'LongInt').SetInt( 15);
25231:   'btVariant', 'LongInt').SetInt( 16);
25232:   'btS64', 'LongInt').SetInt( 17);
25233:   'btU64', 'LongInt').SetInt( 30);
25234:   'btChar', 'LongInt').SetInt( 18);
25235:   'btWideString', 'LongInt').SetInt( 19);
25236:   'btWideChar', 'LongInt').SetInt( 20);
25237:   'btProcPtr', 'LongInt').SetInt( 21);
25238:   'btStaticArray', 'LongInt').SetInt( 22);
25239:   'btSet', 'LongInt').SetInt( 23);
25240:   'btCurrency', 'LongInt').SetInt( 24);

```

```

25241: 'btClass','LongInt').SetInt( 25);
25242: 'btInterface','LongInt').SetInt( 26);
25243: 'btNotificationVariant','LongInt').SetInt( 27);
25244: 'btUnicodeString','LongInt').SetInt( 28);
25245: 'btType','LongInt').SetInt( 130);
25246: 'btEnum','LongInt').SetInt( 129);
25247: 'btExtClass','LongInt').SetInt( 131);
25248: 'CM_A','LongInt').SetInt( 0);
25249: 'CM_CA','LongInt').SetInt( 1);
25250: 'CM_P','LongInt').SetInt( 2);
25251: 'CM_pv','LongInt').SetInt( 3);
25252: 'CM_PO','LongInt').SetInt( 4);
25253: 'Cm_C','LongInt').SetInt( 5);
25254: 'Cm_G','LongInt').SetInt( 6);
25255: 'Cm(CG','LongInt').SetInt( 7);
25256: 'Cm_CNG','LongInt').SetInt( 8);
25257: 'Cm_R','LongInt').SetInt( 9);
25258: 'Cm_ST','LongInt').SetInt( 10);
25259: 'Cm_Pt','LongInt').SetInt( 11);
25260: 'CM_CO','LongInt').SetInt( 12);
25261: 'Cm_cv','LongInt').SetInt( 13);
25262: 'cm_sp','LongInt').SetInt( 14);
25263: 'cm_bn','LongInt').SetInt( 15);
25264: 'cm_vn','LongInt').SetInt( 16);
25265: 'cm_sf','LongInt').SetInt( 17);
25266: 'cm_fg','LongInt').SetInt( 18);
25267: 'cm_puevh','LongInt').SetInt( 19);
25268: 'cm_poehh','LongInt').SetInt( 20);
25269: 'cm_in','LongInt').SetInt( 21);
25270: 'cm_spc','LongInt').SetInt( 22);
25271: 'cm_inc','LongInt').SetInt( 23);
25272: 'cm_dec','LongInt').SetInt( 24);
25273: 'cm_nop','LongInt').SetInt( 255);
25274: 'Cm_PG','LongInt').SetInt( 25);
25275: 'Cm_P2G','LongInt').SetInt( 26);
25276: CL.AddTypeS('TbtU8', 'Byte');
25277: CL.AddTypeS('TbtS8', 'ShortInt');
25278: CL.AddTypeS('TbtU16', 'Word');
25279: CL.AddTypeS('TbtS16', 'SmallInt');
25280: CL.AddTypeS('TbtU32', 'Cardinal');
25281: CL.AddTypeS('TbtS32', 'Longint');
25282: CL.AddTypeS('TbtSingle', 'Single');
25283: CL.AddTypeS('TbtDouble', 'double');
25284: CL.AddTypeS('TbtExtended', 'Extended');
25285: CL.AddTypeS('tbtCurrency', 'Currency');
25286: CL.AddTypeS('tbts64', 'int64');
25287: CL.AddTypeS('Tbtu64', 'uint64');
25288: CL.AddTypeS('TbtString', 'string');
25289: Function MakeHash( const s : TbtString ) : Longint;
25290: //   TbtString = {$IFDEF DELPHI2009UP}AnsiString{$ELSE}String{$ENDIF};
25291: //{$IFDEF PointerSize} 'PointerSize','LongInt').SetInt( IPointer ( 8 4 ));
25292: end;
25293:
25294: -----
25295: mapX:
25296: if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
25297:   writeln('cologne map found');
25298:   GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
25299:   writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'));
25300: OpenMapX('church trier');
25301: GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
25302: writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));
25303: >>> //latitude: '50.94133705' longitude: '6.95812076100766'
25304: // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
25305: writeln(GetGeoCoord('xml','church cefalu sicily',true));
25306: CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TPos): string;
25307: Function SendInput( cInputs: UINT; var pInputs : TInput; cbSize : Integer ) : UINT';
25308: Function GetLastInputInfo( var plii : TLastInputInfo ) : BOOL';
25309: Procedure JvErrorIntercept';
25310: writeln(GetGeoInfo4('178.196.192.131', UrlGeoLookupInfo3));
25311:
25312:
25313: maxbox Ref:
25314: Signature:
25315: SHA1: maxbox3.exe F0AB7D054111F5CE46BA122D6280397A841C6FAB
25316: CRC32: maxbox3.exe 602A885C
25317:
25318: Ref:
25319: 1. writeln(SHA1(Exepath+'\maxbox3.exe'))
25320: 2. shdig: TSHA1Digest;
25321:   shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
25322:     for i:= 0 to 19 do write(Bytetohex(shdig[i]));
25323:
25324: 3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
25325:
25326:
25327: https://www.virustotal.
com/en/file/c2ccaafe689c56d4201ad18bbcf2007ec8cb6e5f86d006028aee2709a311b91d/analysis/1431472027/
25328:

```

```
25329: VirusTotal metadata
25330: -----
25331: First submission 2015-05-12 23:00:59 UTC ( 22 hours, 2 minutes ago )
25332: Last submission 2015-05-12 23:07:07 UTC ( 21 hours, 56 minutes ago )
25333: File names Surprise mx4
25334: maxbox3.exe
25335: maxbox3_9.exe
25336:
25337: SHA256: c2ccaafe689c56d4201ad18bbcf2007ec8cb6e5f86d006028aee2709a311b91d
25338: File name: maxbox3.exe
25339: Detection ratio: 0 / 56
25340: Analysis date: 2015-05-12 23:07:07 UTC ( 21 hours, 55 minutes ago )
25341:
25342: MD5 f08ede9627c8434d35e04ff56e529450
25343: SHA1 f0ab7d054111f5ce46ba122d6280397a841c6fab
25344: SHA256 c2ccaafe689c56d4201ad18bbcf2007ec8cb6e5f86d006028aee2709a311b91d
25345: ssdeep 393216:AUllw/LSA+4sMC/JQUNDIF3lMr6JmDF32J:lw/LSd/ivZJ
25346: authentihash 0401f343b6a009400ae2f3d2d7c0661729962da3956ab9e339c5610427blald7
25347: imphash 2c4488ddale3c370f9740c2c7f22e2b8
25348: File size 24.0 MB ( 25152000 bytes )
25349: File type Win32 EXE
25350: Magic literal
25351: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
25352:
25353: TrID Windows ActiveX control (36.4%)
25354: Inno Setup installer (34.3%)
25355: InstallShield setup (13.4%)
25356: Win32 EXE PECompact compressed (generic) (13.0%)
25357: Win32 Executable (generic) (1.4%)
25358:
25359: ExifTool file metadata
25360: -----
25361:
25362: SpecialBuild mx4
25363: LegalTrademarks maxbox
25364: SubsystemVersion 4.0
25365: Comments reduce to the max
25366: LinkerVersion 2.25
25367: ImageVersion 0.0
25368: FileSubtype 0
25369: FileVersionNumber 3.9.9.195
25370: LanguageCode German (Swiss)
25371: FileFlagsMask 0x003f
25372: FileDescription maxbox Delphi VM
25373: CharacterSet Windows, Latin1
25374: InitializedDataSize 5472768
25375: FileOS Win32
25376: Timestamp 2015:05:12 22:08:10+01:00
25377: MIMEType application/octet-stream
25378: LegalCopyright Free Pascal Script
25379: FileVersion 3.9.9.195
25380: SpeziellesBuild mx4 Compiler Engine
25381: FileType Win32 EXE
25382: PEType PE32
25383: InternalName Surprise mx4
25384: ProductVersion 3.9 Solar mx4
25385: UninitializedDataSize 0
25386: OSVersion 4.0
25387: OriginalFilename maxbox3_9.exe
25388: Subsystem Windows GUI
25389: MachineType Intel 386 or later, and compatibles
25390: CompanyName kleiner kommunikation
25391:
25392: CodeSize 19678208
25393: ProductName maxbox
25394: ProductVersionNumber 3.9.9.195
25395: EntryPoint 0x12c60b0
25396: ObjectFileType Executable application
25397:
25398: ExifTool file metadata
25399: -----
25400: Developer metadata
25401: Publisher kleiner kommunikation
25402: Product maxbox
25403: Original name maxbox3_9.exe
25404: Internal name Surprise mx4
25405: File version 3.9.9.195
25406: Description maxbox Delphi VM
25407: Comments reduce to the max
25408:
25409: PE header basic information
25410:
25411: Target machine Intel 386 or later processors and compatible processors
25412: Compilation timestamp 2015-05-12 21:08:10
25413: Link date 10:08 PM 5/12/2015
25414: Entry Point 0x012C60B0
25415: Number of sections 10
25416:
25417: 10 PE sections
```

```

25418:
25419: Name Virtual address Virtual size Raw size Entropy MD5
25420: .text 4096 19628308 19628544 6.60 3b26cba0ce38b8c057d45ad84384072c
25421: .itext 19636224 49652 49664 6.60 9141fa3ea9a8a714a3d326cadbe20014
25422: .data 19689472 333556 333824 5.62 6ca6e99fc0d042f5c3e9a8ffb0f51f1
25423: .bss 20025344 385568 0.00 d41d8cd98f00b204e9800998ecf8427e
25424: .idata 20414464 58112 58368 5.62 80e9b2e2c89cbad47ec17884f8361a0d
25425: .edata 20475904 77 512 0.90 ee758732620dcf83f0f65dda98a387b1
25426: .tls 20480000 216 0.00 d41d8cd98f00b204e9800998ecf8427e
25427: .rdata 20484096 24 512 0.27 422aa921504e96668acd9676bf246a9f
25428: .reloc 20488192 1251180 1251328 6.76 29c0e0a70410ce57514a9c3a7ad4903c
25429: .rsrc 21741568 3828224 3828224 5.38 7f3a42dbc5e62c07f3017151e47123ce
25430:
25431: PE exports CreateIncome
25432:
25433: Number of PE resources by type
25434: RT_BITMAP 977
25435: RT_STRING 309
25436: RT_RCDATA 165
25437: RT_ICON 56
25438: RT_GROUP_ICON 45
25439: RT_CURSOR 33
25440: RT_GROUP_CURSOR 28
25441: WAVE 9
25442: RT_DIALOG 2
25443: RT_HTML 2
25444: RT_MESSAGEGETABLE 1
25445: RT_MANIFEST 1
25446: AVI 1
25447: RT_VERSION 1
25448:
25449: PE imports
25450: [+] AVICAP32.DLL [+]
25451: [+] IMAGEHLP.DLL [+]
25452: [+] OpenGL32.dll [+]
25453: [+] advapi32.dll [+]
25454: [+] gdi32.dll [+]
25455: [+] iphlpapi.dll [+]
25456: [+] msacm32.dll [+]
25457: [+] oleaut32.dll [+]
25458: [+] shell32.dll [+]
25459: [+] usp10.dll [+]
25460: [+] wininet.dll [+]
25461: [+] ws2_32.dll [+]
25462: [+] wsock32.dll [+]
25463: [+] msimg32.dll [+]
25464: [+] netapi32.dll [+]
25465: Ref:
25466: https://www.virustotal.
com/en/file/c2ccaafe689c56d4201ad18bbcf2007ec8cb6e5f86d006028aee2709a311b91d/analysis/1431472027/
25467: //Commonly used Delphi WinAPI routines
25468: http://www.rosseeld.be/DRO/Delphi/Delphi%20WinAPI.htm
25469: ****
25470: Release Notes maxbox 3.9.9.195 Mai 2015 CODEsign
25471: ****
25472: Add 36 Units, 1 Tutor, SOAPConn, AVI Res, OLEUtils, Wave ACM
25473: Refactor DB Constructor - Virtual Constructor - Override
25474:
25475: 1085 unit uPSI_JvAnimate; //JCL
25476: 1086 unit uPSI_DBXCharDecoder; //DBX
25477: 1087 unit uPSI_JvDBLists; //JCL
25478: 1088 unit uPSI_JvFileInfo; //JCL
25479: 1089 unit uPSI_SOAPConn; //VCL
25480: 1090 unit uPSI_SOAPLinked; //VCL
25481: 1091 unit uPSI_XSBuiltIns; //VCL
25482: 1092 unit uPSI_JvgDigits; //JCL
25483: 1093 unit uPSI_JvDesignUtils;
25484: 1094 unit uPSI_JvgCrossTable;
25485: 1095 unit uPSI_JvgReport; 1096 unit uPSI_JvDBRichEdit;
25486: 1097 unit uPSI_JvWinHelp;
25487: 1098 unit uPSI_WaveConverter; 1099 unit uPSI_ACMConvertor;
25488: 1100 unit XSBuiltIns_Routines;
25489: 1101 unit uPSI_ComObjOleDB_utils.pas;
25490: 1102 unit uPSI_SMScript;
25491: 1103 unit uPSI_CompFileIo;
25492: 1104 unit uPSI_SynHighlighterGeneral;
25493: 1105 unit uPSI_geometry2;
25494: 1106 unit uPSI_MConnect;
25495: 1107 unit uPSI_ObjBrkr;
25496: 1108 unit uPSI_uMultiStr;
25497: 1109 unit uPSI_WinAPI.pas;
25498: 1110 unit uPSI_JvAVICapture;
25499: 1111 unit uPSI_JvExceptionForm;
25500: 1112 unit uPSI_JvConnectNetwork;
25501: 1113 unit uPSI_MTMainForm;
25502: 1114 unit uPSI_DdeMan;
25503: 1115 unit uPSI_DIUtils;
25504: 1116 unit uPSI_gnugettext;
25505: 1117 unit uPSI_Xmlxform;

```

```

25506: 1118 unit uPSI_SvrHTTPIndy;
25507: 1119 unit uPSI_CPortTrmSet;
25508: 1120 unit SvrLog;
25509:
25510: SHA1: maXbox3.exe F0AB7D054111F5CE46BA122D6280397A841C6FAB
25511: CRC32: maXbox3.exe 602A885C
25512:
25513: ****
25514: Release Notes maXbox 3.9.9.180 Feb 2015 CODEsign
25515: ****
25516: Add 20 Units, 1 Slide, Tutor, Big Numbers, TestFramework, GEOInfo
25517:
25518: 1065 unit uPSI_UDict; //DFF
25519: 1066 unit uPSI_ubigFloatV3; //DFF
25520: 1067 unit uPSI_UBigIntsV4; //DFF
25521: 1068 unit uPSI_ServiceMgr2; //mX
25522: 1069 unit uPSI_UP10Build; //PS
25523: 1070 unit uPSI_UParser10; //PS
25524: 1071 unit uPSI_IdModBusServer; //MB
25525: 1072 unit uPSI_IdModBusClient; +MBUtils //MB
25526: 1073 unit uPSI_ColorGrd; //VCL
25527: 1074 unit uPSI_DirOutln; //VCL
25528: 1075 unit uPSI_Gauges; //VCL
25529: 1076 unit uPSI_CustomizeDlg; //VCL
25530: 1077 unit uPSI_ActnMan; //VCL
25531: 1078 unit uPSI_CollPanl; //VCL
25532: 1079 unit uPSI_Calendar2; //VCL
25533: 1080 unit uPSI_IBCtrls; //VCL
25534: 1081 unit uPSI_IdStackWindows; //Indy
25535: 1082 unit uPSI_CTSVendorUtils;
25536: 1083 unit uPSI_VendorTestFramework;
25537: 1084 unit uPSI_TInterval;
25538:
25539: SHA1: maXbox3.exe 3D7F88BE9687CB834A5E2DAED08B23358484FEC0
25540: CRC32: maXbox3.exe E2ADE828
25541:
25542: ****
25543: Release Notes maXbox 3.9.9.160 January 2015 CODEsign
25544: ****
25545: Add 9 Units, 2 Slides 1 Tutor, CLXUp, ExampleEdition, UnitConverter
25546: ExecuteProcess (MultiProcessor), ConsoleCapture (DOS)
25547:
25548: 1053 unit uPSI_BigIni //Hinzen
25549: 1054 unit uPSI_ShellCtrls; //VCL
25550: 1055 unit uPSI_fMath; //FMath
25551: 1056 unit uPSI_fComp; //FMath
25552: 1057 unit uPSI_HighResTimer; //Lauer
25553: 1058 unit uconvMain; (Unit Converter) //PS
25554: 1059 unit uPSI_uconvMain; //PS
25555: 1060 unit uPSI_ParserUtils; //PS
25556: 1061 unit uPSI_upSUtils; //PS
25557: 1062 unit uPSI_ParserU; //PS
25558: 1063 unit uPSI_TypeInfo; //VCL
25559: 1064 unit uPSI_ServiceMgr; //mX
25560:
25561: ****
25562: Release Notes maXbox 3.9.9.120 December 2014 CODEsign
25563: ****
25564: Add 10 Units, 1Slides, NeuralNetwork, Pan3D View, GDIBackend
25565:
25566: 1043 unit uPSI_NeuralNetwork;
25567: 1044 unit uPSI_StExpr;
25568: 1045 unit uPSI_GR32_Geometry;
25569: 1046 unit uPSI_GR32_Containers;
25570: 1047 unit uPSI_GR32_Backends_VCL,
25571: 1048 unit uPSI_StSaturn; //Venus+Pluto+Mars+Merc+JupSat+++
25572: 1049 unit uPSI_JclParseUses;
25573: 1050 unit uPSI_JvFinalize;
25574: 1051 unit uPSI_panUnit1;
25575: 1052 unit uPSI_DD83ul; //Arduino Tester
25576:
25577:
25578: Published Doc maXbox Tutors Starter Introduction 2014
25579: Tutorial 00 Blix the Programmer
25580: Tutorial 01 Procedural-Coding
25581: Tutorial 02 OO-Coding
25582: Tutorial 03 Modular Coding
25583: Tutorial 04 UML Coding
25584: Tutorial 05 Internet Coding
25585: Tutorial 06 Network Coding
25586: Tutorial 07 Game Coding
25587: Tutorial 08 Operating System Coding
25588: Tutorial 09 Database Coding
25589: Tutorial 10 Statistic Coding
25590: Tutorial 11 Forms Coding
25591: Tutorial 12 SQL Coding
25592: Tutorial 13 Crypto Coding
25593: Tutorial 14 Parallel Coding
25594: Tutorial 15 Serial Coding

```

```

25595: Tutorial 16 Event Driven Coding
25596: Tutorial 17 Web Server Coding
25597: Tutorial 18 Arduino Coding and Web of Things
25598: Tutorial 18_3 Arduino RGB LED Breadboard and Source LED Zip
25599: Tutorial 19 WinCOM /Arduino Coding and Source LED COM
25600: Tutorial 20_1 Regular Expressions V2
25601: Tutorial 21 Android av. End of 2014 and Basta LED Things and Code ADK SeekBar
25602: Tutorial 22 Services Coding
25603: Tutorial 23 Real Time Code
25604: Tutorial 24 Clean Code
25605: Tutorial 25 Configuration
25606: Tutorial 26 TCP Sockets
25607: Tutorial 27 XML Coding
25608: Tutorial 28 DLL Coding
25609: Tutorial 29 UML Modeling
25610: Tutorial 30 WOT Web of Things and Basta 2014 Arduino and maXbox
25611: Tutorial 31 Closures
25612: Tutorial 32 SQL Server Firebird
25613: Tutorial 33 Oscilloscope
25614: Tutorial 34 GPS Codes
25615: Tutorial 35 Web Box
25616: Tutorial 36 Function Testing
25617: Tutorial 37 API Coding
25618: Tutorial 39 Maps Coding
25619: Tutorial 39 Maps2 Coding
25620:
25621: http://www.slideshare.net/maxkleiner1/codesign-2015
25622: http://basta_2015_speaker_max_kleiner
25623: http://www.arduino.cc/en/Tutorial/ASCIITable
25624: http://www.vwlowen.co.uk/arduino/usb-digital/pc-control.htm
25625: http://www.yunqa.de/delphi/doku.php/products/regex/syntax#quantifiers
25626: http://elib.uni-stuttgart.de/opus/volltexte/2008/3440/pdf/diss_kroell_hs.pdf
25627:
25628:
25629:
25630: Published Doc maXbox Example Edition 2015
25631:
25632: \example_edition\01_Algorithm';
25633: \example_edition\02_Graphics';
25634: \example_edition\03_Games';
25635: \example_edition\04_Multimedia';
25636: \example_edition\05_Internet';
25637: \example_edition\06_Communication';
25638: \example_edition\07_Geographical';
25639: \example_edition\08_Operating';
25640: \example_edition\09_Database';
25641: \example_edition\10_Science';
25642: \example_edition\11_EMBEDDED';
25643: \example_edition\12_Security';
25644: \example_edition\13_General';
25645: \example_edition\14_Energy';
25646: \example_edition\15_Transport';

25647:
25648:
25649: Ref:
25650: if IsValidPeFile(exepath+'maxbox3.exe') then begin
25651:   x1:= ComputePEChecksum(exepath+'maxbox3.exe'); // original filename
25652:   x2:= ComputePEChecksum(exepath+'maxbox3.exe');
25653: end;
25654: WriteLn('Checksum 1: '+ itoa(x1)+ #13#10+'Checksum 2: '+ itoa(x2));
25655:
25656: if ConnectDrive('Z:','\\MAXBOX8\Users\Public', True,True) = NO_ERROR then
25657:   writeln('Net Share Z:\ Connected');
25658:   DisconnectNetDrive('Z:', True, True, True);
25659:
25660: ComTerminal:= TCustomComTerminal.create(self);
25661:   EditComTerminal(comterminal); //TComTrnSetForm
25662: comterminal.Free;
25663:
25664: LastSysErrorMessage : string; LastSysErrorMessageA : AnsiString';
25665:
25666:
25667: SHA1: maxbox3.exe F0AB7D054111F5CE46BA122D6280397A841C6FAB
25668: CRC32: maxbox3.exe 602A885C
25669:
25670: ---- bigbitbox code_cleared_checked----

```