

```

1: ****
2: Constructor Function and Procedure List of maxbox 3.9.9.190
3: ****
4:
5: //////////////////////////////////////////////////////////////////
6: ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX3
7: -----
8:
9: File EXE: 24496640 V3.9.9.190 March 2015 to EKON/BASTA/JAX/IBZ/SWS/EU/DT/PASCON
10: *****Now the Funclist*****
11: Funclist Function : 14417 //10766 //10165 (7648)
12: *****Now the Proclist*****
13: Proclist Procedure Size is: 8976 //6792 //6401 4752
14: *****Now Constructors*****
15: Constructlist Constructor Size is: 1458 //995 //
16: def head:max: maxbox7: 14.03.2015 12:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: -----
20: Funclist total Size all is: 24851! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 24860
22: ASize of EXE: 24496640 (23614464) (22041600) (16586240) (13511680)
23: SHA1 Hash of maxbox 3.9.9.190: A5C1A64E94EBC01D01B67DB35C3E27E50607DFFB
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 DelimiterPosnL( 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 AType : 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; PasswordChar : 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 GetDFAState( oStates : TList ) : TniRegularExpressionState
1088: Function GetDirectorySize( const Path : string ) : Int64
1089: Function GetDisplayWidth : Integer
1090: Function GetDLLVersion( const DLLName : string; var pdwMajor, pdwMinor : Integer ) : Boolean
1091: Function GetDomainName : string
1092: Function GetDriverRegistryFile( DesignMode : Boolean ) : string
1093: function GetDriveType(rootpath: pchar): cardinal;
1094: Function GetDriveTypeStr( const Drive : Char ) : string
1095: Function GetEnumerator : TFavoriteLinkItemsEnumerator
1096: Function GetEnumerator : TListItemsEnumerator
1097: Function GetEnumerator : TTaskDialogButtonsEnumerator
1098: Function GetEnumerator : TToolBarEnumerator
1099: Function GetEnumerator : TTreeNodesEnumerator
1100: Function GetEnumerator : TWideStringsEnumerator
1101: Function GetEnvVar( const VarName : string ) : string
1102: Function GetEnvironmentVar( const AVariableName : string ) : string
1103: Function GetEnvironmentVariable( const VarName : string ) : string
1104: Function GetEnvironmentVar( const Name : string; var Value : string; Expand : Boolean ) : Boolean
1105: Function GetEnvironmentVars( const Vars : TStrings; Expand : Boolean ) : Boolean
1106: Function getEnvironmentString: string;
1107: Function GetExceptionHandler : TObject
1108: Function GetFavoritesFolder : string
1109: Function GetFieldByName( const Name : string ) : string
1110: Function GetFieldInfo( const Origin : Widestring; var FieldInfo : TFieldInfo ) : Boolean
1111: Function GetFieldValue( ACol : Integer ) : string
1112: Function GetFileAgeCoherence( const FileName : string ) : Boolean
1113: Function GetFileCreation( const FileName : string ) : TfileTime
1114: Function GetFileCreationTime( const Filename : string ) : TDateTime
1115: Function GetFileInfo( const FileName : string ) : TSearchRec
1116: Function GetFileLastAccess( const FileName : string ) : TfileTime
1117: Function GetFileLastWrite( const FileName : string ) : TfileTime
1118: Function GetFileList(FileList: TStringlist; apath: string): TStringlist;
1119: Function GetFileList1(apath: string): TStringlist;
1120: Function GetFileMIMEType( const AFileName : string ) : string
1121: Function GetFileSize( const FileName : string ) : Int64
1122: Function GetFileVersion( AFileName : string ) : Cardinal
1123: Function GetFileVersion( const AFilename : string ) : Cardinal
1124: Function GetFileVersion2(Handle: Integer; x: Integer): Integer; stdcall;
1125: Function GetFileDate(aFile:string; aWithTime:Boolean):string;
1126: Function GetFileCount(adirmask: string): integer; //files count in directory!
1127: Function GetFilterData( Root : PExprNode ) : TExprData
1128: Function getFirstChild : TTreeNode
1129: Function getFirstChild : LongInt
1130: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string ) : string
1131: Function GetFirstNode : TTreeNode
1132: Function GetFontsFolder : string
1133: Function GetFormulaValue( const Formula : string ) : Extended
1134: Function GetFreePageFileMemory : Integer
1135: Function GetFreePhysicalMemory : Integer
1136: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind ) : Integer;
1137: Function GetFreeSystemResources1 : TFreeSystemResources;
1138: Function GetFreeVirtualMemory : Integer

```

```

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

```

```

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

```

```

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

```

```

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

```

```

1489: Function IBFormatIdentifierValue( Dialect : Integer; Value : String ) : String
1490: Function IBExtractIdentifier( Dialect : Integer; Value : String ) : String
1491: Function IBQuoteIdentifier( Dialect : Integer; Value : String ) : String
1492: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1493: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1494: Function RandomString( iLength : Integer ) : String';
1495: Function RandomInteger( iLow, iHigh : Integer ) : Integer';
1496: Function StripString( st : String; CharsToStrip : String ) : String';
1497: FUNCTION 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( RtlSignum : 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(hdwnd: 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 StreamToDecimalbytes( 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 TaskMessageDlgl( 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: function TimeStampToMSECS(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(S: WideString): WideString;
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 vwsprintf( 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, erInterfaceNotSupportedError);
3327: (*-----*)
3328: procedure SIRегистер_JclIniFiles(CL: TPSPPascalCompiler);
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: TPSPPascalCompiler);
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: Function StrToken( var S : AnsiString; Separator : AnsiChar ) : AnsiString
3602: Procedure StrTokens( const S : AnsiString; const List : TStrings )
3603: Procedure StrTokenToStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings )
3604: //Function StrWord( var S : PAnsiChar; out Word : AnsiString ) : Boolean
3605: Function StrToFloatSafe( const S : AnsiString ) : Float
3606: Function StrToIntSafe( const S : AnsiString ) : Integer
3607: Procedure StrNormIndex( const Strlen : Integer; var Index : Integer; var Count : Integer );
3608: Function ArrayOf( List : TStrings ) : TDynStringArray;
3609:   EJclStringError', 'EJclError
3610: function IsClass(Address: TObject): Boolean;
3611: function IsObject(Address: TObject): Boolean;
3612: // Console Utilities
3613: //function ReadKey: Char;
3614: function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3615: function JclGUIDToString(const GUID: TGUID): string;
3616: function JclStringToGUID(const S: string): TGUID;
3617: end;
3618:
3619:
3620: ****uPSI_JvDBUtil;
3621: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
3622: Function GetQueryResult( const DatabaseName, SQL : string ) : Variant
3623: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant; const AResultName : string ) : Variant
3624: //Function StrFieldDesc( Field : FLDDesc ) : string
3625: Function Var2Type( V : Variant; const VarType : Integer ) : Variant
3626: Procedure CopyRecord( DataSet : TDataSet )
3627: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string; MasterField : Word; ModOp, DelOp : RINTQual )
3628: Procedure AddMasterPassword( Table : TTable; pswd : string )
3629: Procedure PackTable( Table : TTable )
3630: Procedure PackEncryptedTable( Table : TTable; pswd : string )
3631: Function EncodeQuotes( const S : string ) : string
3632: Function Cmp( const S1, S2 : string ) : Boolean
3633: Function SubStr( const S : string; const Index : Integer; const Separator : string ) : string
3634: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string ) : string
3635: Function ReplaceString( S : string; const OldPattern, NewPattern : string ) : string
3636: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer )
3637: *****uPSI_JvJvBDEUtils;*****
3638: //JvBDEUtils
3639: Function CreateDbLocate : TJvLocateObject
3640: //Function CheckOpen( Status : DBTResult ) : Boolean
3641: Procedure FetchAllRecords( DataSet : TBDEDataSet )
3642: Function TransActive( Database : TDatabase ) : Boolean
3643: Function AsyncQrySupported( Database : TDatabase ) : Boolean
3644: Function GetQuoteChar( Database : TDatabase ) : string
3645: Procedure ExecuteQuery( const DbName, QueryText : string )
3646: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string )
3647: Function FieldLogicMap( FldType : TFieldType ) : Integer
3648: Function FieldsSubtypeMap( FldType : TFieldType ) : Integer Value : string; Buffer : Pointer )
3649: Function GetAliasPath( const AliasName : string ) : string
3650: Function IsDirectory( const DatabaseName : string ) : Boolean
3651: Function GetBdeDirectory : string
3652: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent ) : Boolean
3653: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value,FieldName : string ) : Boolean
3654: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value,FieldName : string ) : Boolean
3655: Function DataSetRecNo( DataSet : TDataSet ) : Longint
3656: Function DataSetRecordCount( DataSet : TDataSet ) : Longint
3657: Function DataSetPositionStr( DataSet : TDataSet ) : string
3658: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean )
3659: Function CurrentRecordDeleted( DataSet : TBDEDataSet ) : Boolean
3660: Function IsFilterApplicable( DataSet : TDataSet ) : Boolean
3661: Function IsBookmarkStable( DataSet : TBDEDataSet ) : Boolean
3662: Procedure SetIndex( Table : TTable; const IndexFieldNames : string )
3663: Procedure RestoreIndex( Table : TTable )
3664: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const )
3665: Procedure PackTable( Table : TTable )
3666: Procedure ReindexTable( Table : TTable )
3667: Procedure BdeFlushBuffers

```

```

3668: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer ) : Pointer
3669: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string )
3670: Procedure DbNotSupported
3671: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
3672: AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint )
3673: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
3674: AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter:Char;MaxRecordCount:Longint );
3675: ****uPSI_JvDateUtil;
3676: function CurrentYear: Word;
3677: function IsLeapYear(AYear: Integer): Boolean;
3678: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3679: function FirstDayOfPrevMonth: TDateTime;
3680: function LastDayOfPrevMonth: TDateTime;
3681: function FirstDayOfNextMonth: TDateTime;
3682: function ExtractDay(ADate: TDateTime): Word;
3683: function ExtractMonth(ADate: TDateTime): Word;
3684: function ExtractYear(ADate: TDateTime): Word;
3685: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3686: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3687: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3688: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3689: function ValidDate(ADate: TDateTime): Boolean;
3690: procedure DateDiff(Datel, Date2: TDateTime; var Days, Months, Years: Word);
3691: function MonthsBetween(Datel, Date2: TDateTime): Double;
3692: function DaysInPeriod(Datel, Date2: TDateTime): Longint;
3693: { Count days between Datel and Date2 + 1, so if Datel = Date2 result = 1 }
3694: function DaysBetween(Datel, Date2: TDateTime): Longint;
3695: { The same as previous but if Date2 < Datel result = 0 }
3696: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3697: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3698: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3699: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3700: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3701: function CutTime(ADate: TDateTime): { Set time to 00:00:00:00 }
3702: { String to date conversions }
3703: function GetDateOrder(const DateFormat: string): TDateOrder;
3704: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3705: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3706: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3707: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3708: function DefDateFormat(FourDigitYear: Boolean): string;
3709: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3710: -----
3711: ***** JvUtils*****
3712: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3713: function GetWordOnPos(const S: string; const P: Integer): string;
3714: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3715: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3716: { SubStr returns substring from string, S, separated with Separator string}
3717: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3718: { SubStrEnd same to previous function but Index numerated from the end of string }
3719: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3720: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3721: function SubWord(P: PChar; var P2: PChar): string;
3722: { NumberByWord returns the text representation of
3723:   the number, N, in normal russian language. Was typed from Monitor magazine }
3724: function NumberByWord(const N: Longint): string;
3725: // function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3726: //the symbol Pos is pointed. Lines separated with #13 symbol
3727: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3728: { GetXYByPos is same to previous function, but returns X position in line too}
3729: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3730: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3731: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3732: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3733: function ConcatSep(const S, S2, Separator: string): string;
3734: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3735: function ConcatLeftSep(const S, S2, Separator: string): string;
3736: { MinimizeString trunks long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3737: function MinimizeString(const S: string; const MaxLen: Integer): string;
3738: { Next 4 function for russian chars transliterating.
3739:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3740: procedure Dos2Win(var S: string);
3741: procedure Win2Dos(var S: string);
3742: function Dos2WinRes(const S: string): string;
3743: function Win2DosRes(const S: string): string;
3744: function Win2Koi(const S: string): string;
3745: { Spaces returns string consists on N space chars }
3746: function Spaces(const N: Integer): string;
3747: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3748: function AddSpaces(const S: string; const N: Integer): string;
3749: { function LastDate for russian users only } { returns date relative to current date: '' }
3750: function LastDate(const Dat: TDateTime): string;
3751: { CurrencyToStr format currency, Cur, using ffcurrency float format}
3752: function CurrencyToStr(const Cur: currency): string;
3753: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}

```

```

3754: function Cmp(const S1, S2: string): Boolean;
3755: { StringCat add S2 string to S1 and returns this string }
3756: function StringCat(var S1: string; S2: string): string;
3757: { HasChar returns True, if Char, Ch, contains in string, S }
3758: function HasChar(const Ch: Char; const S: string): Boolean;
3759: function HasAnyChar(const Chars: string; const S: string): Boolean;
3760: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3761: function CountOfChar(const Ch: Char; const S: string): Integer;
3762: function DefStr(const S: string; Default: string): string;
3763: {**** files routines}
3764: { GetWinDir returns Windows folder name }
3765: function GetWinDir: TFileName;
3766: function GetSysDir: String;
3767: { GetTempDir returns Windows temporary folder name }
3768: function GetTempDir: string;
3769: { GetTempFileName returns temporary file name on drive, there FileName is placed }
3770: function GenTempFileName(FileName: string): string;
3771: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3772: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3773: { ClearDir clears folder Dir }
3774: function ClearDir(const Dir: string): Boolean;
3775: { DeleteDir clears and than delete folder Dir }
3776: function DeleteDir(const Dir: string): Boolean;
3777: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3778: function FileEquMask(FileName, Mask: TFileName): Boolean;
3779: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3780:   Masks must be separated with comma (';') }
3781: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3782: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3783: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3784: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3785: { FileGetInfo fills SearchRec record for specified file attributes}
3786: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3787: { HasSubFolder returns True, if folder APath contains other folders }
3788: function HasSubFolder(APath: TFileName): Boolean;
3789: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3790: function IsEmptyFolder(APath: TFileName): Boolean;
3791: { AddSlash add slash Char to Dir parameter, if needed }
3792: procedure AddSlash(var Dir: TFileName);
3793: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3794: function AddSlash2(const Dir: TFileName): string;
3795: { AddPath returns FileName with Path, if FileName not contain any path }
3796: function AddPath(const FileName, Path: TFileName): TFileName;
3797: function AddPaths(const PathList, Path: string): string;
3798: function ParentPath(const Path: TFileName): TFileName;
3799: function FindInPath(const FileName, PathList: string): TFileName;
3800: function FindinPaths(const fileName,paths: String): String;
3801: {$IFDEF BCB1}
3802: { BrowseForFolder displays Browse For Folder dialog }
3803: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3804: {$ENDIF BCB1}
3805: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
  AHelpContext : THelpContext) : Boolean;
3806: Function BrowseForComputer(const Atitle : string; AllowCreate : Boolean; var ADirectory : string;
  AHelpContext : THelpContext) : Boolean;
3807: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean;
3808: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean;
3809:
3810: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3811: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3812: { HasParam returns True, if program running with specified parameter, Param }
3813: function HasParam(const Param: string): Boolean;
3814: function HasSwitch(const Param: string): Boolean;
3815: function Switch(const Param: string): string;
3816: { ExePath returns ExtractFilePath(ParamStr(0)) }
3817: function ExePath: TFileName;
3818: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3819: function FileTimeToDate(FT: TFileTime): TDateTime;
3820: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3821: {**** Graphic routines }
3822: { TTFontSelected returns True, if True Type font is selected in specified device context }
3823: function TTFontSelected(const DC: HDC): Boolean;
3824: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3825: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3826: {**** Windows routines }
3827: { SetWindowTop put window to top without recreating window }
3828: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3829: {**** other routines }
3830: { KeyPressed returns True, if Key VK is now pressed }
3831: function KeyPressed(VK: Integer): Boolean;
3832: procedure SwapInt(var Int1, Int2: Integer);
3833: function IntPower(Base, Exponent: Integer): Integer;
3834: function ChangeTopException(E: TObject): TObject;
3835: function StrToBool(const S: string): Boolean;
3836: {$IFDEF COMPILER3_UP}
3837: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3838:   Length of MaxLen bytes. The compare operation is controlled by the
3839:   current Windows locale. The return value is the same as for CompareStr. }
3840: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;

```

```

3841: function AnsiStrICmp(S1, S2: PChar): Integer;
3842: {$ENDIF}
3843: function Var2Type(V: Variant; const VarType: Integer): Variant;
3844: function VarToInt(V: Variant): Integer;
3845: function VarToFloat(V: Variant): Double;
3846: { following functions are not documented because they are don't work properly , so don't use them }
3847: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3848: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3849: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3850: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3851: function GetParameter: string;
3852: function GetLongFileName(FileName: string): string;
3853: {* from FileCtrl}
3854: function DirectoryExists(const Name: string): Boolean;
3855: procedure ForceDirectories(Dir: string);
3856: {# from FileCtrl}
3857: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3858: function GetComputerID: string;
3859: function GetComputerName: string;
3860: {**** string routines }
3861: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the same Index.Also see RAUtilsW.ReplaceSokr1 function }
3862: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3863: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3864:   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 }
3865: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3866: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3867: function CountOfLines(const S: string): Integer;
3868: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3869: procedure DeleteEmptyLines(Ss: TStrings);
3870: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3871:   Note: If strings SQL already contains where-statement, it must be started on begining of any line }
3872: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3873: {**** files routines - }
3874: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3875:   Resource can be compressed using MS Compress program}
3876: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3877: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool;
3878: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3879: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3880: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3881: { IniReadSection read section, Section, from ini-file,
3882:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3883:   Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
3884: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3885: { LoadTextFile load text file, FileName, into string }
3886: function LoadTextFile(const FileName: TFileName): string;
3887: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3888: { ReadFolder reads files list from disk folder, Folder, that are equal Mask, into strings, FileList}
3889: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3890: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3891: {$IFDEF COMPILER3_UP}
3892: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3893: function TargetFileName(const FileName: TFileName): TFileName;
3894: { return filename ShortCut linked to }
3895: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3896: {$ENDIF COMPILER3_UP}
3897: {**** Graphic routines - }
3898: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3899: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3900: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3901: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3902: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3903: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3904: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3905: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3906: { Cinema draws some visual effect }
3907: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3908: { Roughed fills rect with special 3D pattern }
3909: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3910: { 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 }
3911: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3912: { TextWidth calculate text with for writing using standard desktop font }
3913: function TextWidth(AStr: string): Integer;
3914: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3915: function DefineCursor(Identifier: PChar): TCursor;
3916: {**** other routines - }
3917: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3918: function FindFormByClass(FormClass: TFormClass): TForm;
3919: function FindFormByClassName(FormClassName: string): TForm;
3920: { FindByTag returns the control with specified class, ComponentClass, from WinControl.Controls property,
3921:   having Tag property value, equaled to Tag parameter }
3922: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3923: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3924: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3925: { RBTAG searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3926: function RBTAG(Parent: TWinControl): Integer;

```

```

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

```

```

4012: Procedure JvEatWhitespaceChars1( S1 : PWideChar );
4013: Function GetTokenCount : Integer
4014: Procedure ResetTokenCount
4015: end;
4016:
4017: procedure SJRegister_JvDBQueryParamsForm(CL: TPSPPascalCompiler);
4018: begin
4019:   SJRegister_TJvQueryParamsDialog(CL);
4020:   Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext ) : Boolean
4021: end;
4022:
4023: ***** JvStringUtil / JvStringUtilities *****
4024: function FindNotBlankCharPos(const S: string): Integer;
4025: function AnsiChangeCase(const S: string): string;
4026: function GetWordOnPos(const S: string; const P: Integer): string;
4027: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4028: function Cmp(const S1, S2: string): Boolean;
4029: { Spaces returns string consists on N space chars }
4030: function Spaces(const N: Integer): string;
4031: { HasChar returns True, if char, Ch, contains in string, S }
4032: function HasChar(const Ch: Char; const S: string): Boolean;
4033: function HasAnyChar(const Chars: string; const S: string): Boolean;
4034: { SubStr returns substring from string, S, separated with Separator string}
4035: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
4036: { SubStrEnd same to previous function but Index numerated from the end of string }
4037: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4038: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
4039: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
4040: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
4041: { GetXYByPos is same to previous function, but returns X position in line too}
4042: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4043: { AddSlash returns string with added slash char to Dir parameter, if needed }
4044: function AddSlash2(const Dir: TFileName): string;
4045: { AddPath returns FileName with Path, if FileName not contain any path }
4046: function AddPath(const FileName, Path: TFileName): TFileName;
4047: { ExePath returns ExtractFilePath(ParamStr(0)) }
4048: function ExePath: TFileName;
4049: function LoadTextFile(const FileName: TFileName): string;
4050: procedure SaveTextfile(const FileName: TFileName; const Source: string);
4051: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4052: function ConcatSep(const S, S2, Separator: string): string;
4053: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4054: function FileEquMask(FileName, Mask: TFileName): Boolean;
4055: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4056:   Masks must be separated with comma (';') }
4057: function FileEquMasks(FileName, Masks: TFileName): Boolean;
4058: function StringEndsWith(const Str, SubStr: string): Boolean;
4059: function ExtractFilePath2(const FileName: string): string;
4060: function StrToOem(const AnsiStr: string): string;
4061: { StrToOem translates a string from the Windows character set into the OEM character set. }
4062: function OemToAnsiStr(const OemStr: string): string;
4063: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4064: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4065: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4066: function ReplaceStr(const S, Srch, Replace: string): string;
4067: { Returns string with every occurrence of Srch string replaced with Replace string. }
4068: function DelSpace(const S: string): string;
4069: { DelSpace return a string with all white spaces removed. }
4070: function DelChars(const S: string; Chr: Char): string;
4071: { DelChars return a string with all Chr characters removed. }
4072: function DelBSpace(const S: string): string;
4073: { DelBSpace trims leading spaces from the given string. }
4074: function DelEspace(const S: string): string;
4075: { DelEspace trims trailing spaces from the given string. }
4076: function DelRSpace(const S: string): string;
4077: { DelRSpace trims leading and trailing spaces from the given string. }
4078: function DelSpace1(const S: string): string;
4079: { DelSpace1 return a string with all non-single white spaces removed. }
4080: function Tab2Space(const S: string; Numb: Byte): string;
4081: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4082: function NPos(const C: string; S: string; N: Integer): Integer;
4083: { NPos searches for a N-th position of substring C in a given string. }
4084: function MakeStr(C: Char; N: Integer): string;
4085: function MS(C: Char; N: Integer): string;
4086: { MakeStr return a string of length N filled with character C. }
4087: function AddChar(C: Char; const S: string; N: Integer): string;
4088: { AddChar return a string left-padded to length N with characters c. }
4089: function AddCharR(C: Char; const S: string; N: Integer): string;
4090: { AddCharR return a string right-padded to length N with characters c. }
4091: function LeftStr(const S: string; N: Integer): string;
4092: { LeftStr return a string right-padded to length N with blanks. }
4093: function RightStr(const S: string; N: Integer): string;
4094: { RightStr return a string left-padded to length N with blanks. }
4095: function CenterStr(const S: string; Len: Integer): string;
4096: { CenterStr centers the characters in the string based upon the Len specified. }
4097: function CompStr(const S1, S2: string): Integer;
4098: { CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4099: function CompText(const S1, S2: string): Integer;
4100: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }

```

```

4101: function Copy2Symb(const S: string; Symb: Char): string;
4102: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4103: function Copy2SymbDel(var S: string; Symb: Char): string;
4104: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4105: function Copy2Space(const S: string): string;
4106: { Copy2Space returns a substring of a string S from begining to first white space. }
4107: function Copy2SpaceDel(var S: string): string;
4108: { Copy2SpaceDel returns a substring of a string S from begining to first
4109:   white space and removes this substring from S. }
4110: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4111: { Returns string, with the first letter of each word in uppercase,
4112:   all other letters in lowercase. Words are delimited by WordDelims. }
4113: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4114: { WordCount given a set of word delimiters, returns number of words in S. }
4115: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4116: { Given a set of word delimiters, returns start position of N'th word in S. }
4117: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4118: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TCharSet; var Pos: Integer): string;
4119: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4120: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4121:   delimiters, return the N'th word in S. }
4122: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4123: { ExtractSubstr given set of word delimiters, returns the substring from S, that started from position Pos. }
4124: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4125: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4126: function QuotedString(const S: string; Quote: Char): string;
4127: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4128: function ExtractQuotedString(const S: string; Quote: Char): string;
4129: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4130:   and reduces pairs of Quote characters within the quoted string to a single character. }
4131: function FindPart(const HelpWilds, InputStr: string): Integer;
4132: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4133: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4134: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4135: function XorString(const Key, Src: ShortString): ShortString;
4136: function XorEncode(const Key, Source: string): string;
4137: function XorDecode(const Key, Source: string): string;
4138: { ** Command line routines ** }
4139: {$IFNDEF COMPILER4_UP}
4140: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet; IgnoreCase: Boolean): Boolean;
4141: {$ENDIF}
4142: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4143: { ** Numeric string handling routines ** }
4144: function Numb2USA(const S: string): string;
4145: { Numb2USA converts numeric string S to USA-format. }
4146: function Dec2Hex(N: Longint; A: Byte): string;
4147: function D2H(N: Longint; A: Byte): string;
4148: { Dec2Hex converts the given value to a hexadecimal string representation
4149:   with the minimum number of digits (A) specified. }
4150: function Hex2Dec(const S: string): Longint;
4151: function H2D(const S: string): Longint;
4152: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4153: function Dec2Numb(N: Longint; A, B: Byte): string;
4154: { Dec2Numb converts the given value to a string representation with the
4155:   base equal to B and with the minimum number of digits (A) specified. }
4156: function Numb2Dec(S: string; B: Byte): Longint;
4157: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4158: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4159: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4160: function IntToRoman(Value: Longint): string;
4161: { IntToRoman converts the given value to a roman numeric string representation. }
4162: function RomanToInt(const S: string): Longint;
4163: { RomanToInt converts the given string to an integer value. If the string
4164:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4165: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4166: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4167: ***** JvFileUtil*****
4168: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4169: procedure CopyFileEx(const FileName, DestName: string; OverwriteReadOnly, ShellDialog: Boolean; ProgressControl: TControl);
4170: procedure MoveFile(const FileName, DestName: TFileName);
4171: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4172: {$IFDEF COMPILER4_UP}
4173: function GetFileSize(const FileName: string): Int64;
4174: {$ELSE}
4175: function GetFileSize(const FileName: string): Longint;
4176: {$ENDIF}
4177: function FileDateTime(const FileName: string): TDateTime;
4178: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4179: function DeleteFiles(const FileMask: string): Boolean;
4180: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4181: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4182: function NormalDir(const DirName: string): string;
4183: function RemoveBackSlash(const DirName: string): string;
4184: function ValidFileName(const FileName: string): Boolean;
4185: function DirExists(Name: string): Boolean;
4186: procedure ForceDirectories(Dir: string);
4187: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4188: {$IFDEF COMPILER4_UP} overload; {$ENDIF}

```

```

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

```

```

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

```

```

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

```

```

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

```

```

4541: function GetWordOnPos(const S: string; const P: Integer): string;
4542: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4543: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4544: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4545: { GetWordOnPosEx working like GetWordOnPos function, but
4546:   also returns Word position in iBeg, iEnd variables }
4547: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4548: function GetWordOnPosExW(const S: WideString; const P: Integer): WideString;
4549: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4550: function GetNextWordPosExW(const Text: WideString; StartIndex: Integer; var iBeg, iEnd: Integer): WideString;
4551: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4552: { GetEndPosCaret returns the caret position of the last char. For the position
4553:   after the last char of Text you must add 1 to the returned X value. }
4554: procedure GetEndPosCaretW(const Text: WideString; CaretX, CaretY: Integer; var X, Y: Integer);
4555: { GetEndPosCaret returns the caret position of the last char. For the position
4556:   after the last char of Text you must add 1 to the returned X value. }
4557: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4558: function SubStrBySeparator(const S: string; const Index: Integer; const
4559: Separator: string; startIndex: Int=1): string;
4560: function SubStrBySeparatorW(const S: WideString; const Index: Int; const
4561: Separator: WideString; startIndex: Int): WideString;
4562: { SubStrEnd same to previous function but Index numerated from the end of string }
4563: function SubWord(const S: string; const Index: Integer; const Separator: string): string;
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;
4576: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4577: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
4578: SUPPORTS_INLINE}
4579: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
4580: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
4581: SUPPORTS_INLINE}
4582: { Next 4 function for russian chars transliterating.
4583:   This functions are needed because Oem2ansi and Ansi20em functions sometimes suck }
4584: procedure Dos2Win(var S: AnsiString);
4585: procedure Win2Dos(var S: AnsiString);
4586: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4587: function Win2DOSRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4588: function Win2Koi(const S: AnsiString): AnsiString;
4589: { FillString fills the string Buffer with Count Chars }
4590: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4591: procedure FillString(var Buffer: string; startIndex, Count: Integer; const Value: Char); overload;
4592: { MoveString copies Count Chars from Source to Dest }
4593: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
4594: inline; {$ENDIF SUPPORTS_INLINE} overload;
4595: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4596: DstStartIdx: Integer; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4597: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4598: procedure FillWideChar(var Buffer: WideString; Count: Integer; const Value: WideChar);
4599: { MoveWideChar copies Count WideChars from Source to Dest }
4600: 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 AddSpacesW(const S: WideString; const N: Integer): WideString;
4609: { function LastDateRUS for russian users only }
4610: { returns date relative to current date: 'âââ äïý ïâçââ' }
4611: function LastDateRUS(const Dat: TDateTime): string;
4612: { CurrencyToStr format Currency, Cur, using ffcurrency float format}
4613: function CurrencyToStr(const Cur: Currency): string;
4614: { HasChar returns True, if Char, Ch, contains in string, S }
4615: function HasChar(const Ch: Char; const S: string): Boolean;
4616: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4617: function HasAnyChar(const Chars: string; const S: string): Boolean;
4618: {$IFDEF COMPILER12_UP}
4619: function CharInSet(const Ch: AnsiChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4620: {$ENDIF ~COMPILER12_UP}

```

```

4621: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
4622:   SUPPORTS_INLINE}
4623: function CountOfChar(const Ch: Char; const S: string): Integer;
4624: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
4625:   SUPPORTS_INLINE}
4626: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4627: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4628: function StrPosW(S, SubStr: PWideChar): PWideChar;
4629: function StrLenW(S: PWideChar): Integer;
4630: function TrimW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4631: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4632: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4633: TPixelFormat', '(pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4634: TMappingMethod', '(mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4635: Function GetBitmapPixelFormat( Bitmap : TBitmap ) : TPixelFormat
4636: Function GetPaletteBitmapFormat( Bitmap : TBitmap ) : TPixelFormat
4637: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4638: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4639: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4640: Function ScreenPixelFormat : TPixelFormat
4641: Function ScreenColorCount : Integer
4642: Procedure TileImage( Canvas: TCanvas; Rect : TRect; Image : TGraphic)
4643: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4644: // SJRegister_TJvGradient(CL);
4645:
4646: {***** files routines}
4647: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4648: const
4649:   {$IFDEF MSWINDOWS}
4650:     DefaultCaseSensitivity = False;
4651:   {$ENDIF MSWINDOWS}
4652:   {$IFDEF UNIX}
4653:     DefaultCaseSensitivity = True;
4654:   {$ENDIF UNIX}
4655: { GenTempFileName returns temporary file name on
4656:   drive, there FileName is placed }
4657: function GenTempFileName(FileName: string): string;
4658: { GenTempFileNameExt same to previous function, but
4659:   returning filename has given extension, FileExt }
4660: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4661: { ClearDir clears folder Dir }
4662: function ClearDir(const Dir: string): Boolean;
4663: { DeleteDir clears and than delete folder Dir }
4664: function DeleteDir(const Dir: string): Boolean;
4665: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4666: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4667: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4668:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4669: function FileEquMasks(FileName, Masks: TFileName;
4670:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4671: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4672: {$IFDEF MSWINDOWS}
4673: { LZFileExpand expand file, FileSource,
4674:   into FileDest. Given file must be compressed, using MS Compress program }
4675: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4676: {$ENDIF MSWINDOWS}
4677: { FileInfo fills SearchRec record for specified file attributes}
4678: function FileInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4679: { HasSubFolder returns True, if folder APath contains other folders }
4680: function HasSubFolder(APath: TFileName): Boolean;
4681: { IsEmptyFolder returns True, if there are no files or
4682:   folders in given folder, APPath}
4683: function IsEmptyFolder(APath: TFileName): Boolean;
4684: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4685: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4686: { AddPath returns FileName with Path, if FileName not contain any path }
4687: function AddPath(const FileName, Path: TFileName): TFileName;
4688: function AddPaths(const PathList, Path: string): string;
4689: function ParentPath(const Path: TFileName): TFileName;
4690: function FindInPath(const FileName, PathList: string): TFileName;
4691: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4692: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4693: { HasParam returns True, if program running with specified parameter, Param }
4694: function HasParam(const Param: string): Boolean;
4695: function HasSwitch(const Param: string): Boolean;
4696: function Switch(const Param: string): string;
4697: { ExePath returns ExtractFilePath(ParamStr(0)) }
4698: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4699: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4700: //function FileTimeToDate(FT: TFileTime): TDateTime;
4701: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4702: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4703: {*** Graphic routines }
4704: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4705: function IsTTFontSelected(const DC: HDC): Boolean;
4706: function KeyPressed(VK: Integer): Boolean;

```

```

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

```

```

4793:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4794: function ItemHtPlain(const Text: string): string;
4795: { ClearList - clears list of TObject }
4796: procedure ClearList(List: TList);
4797: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
4798: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4799: { RTTI support }
4800: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4801: function GetPropStr(Obj: TObject; const PropName: string): string;
4802: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4803: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4804: procedure PrepareIniSection(Ss: TStrings);
4805: { following functions are not documented because they are don't work properly, so don't use them }
4806: // (rom) from JVBandWindows to make it obsolete
4807: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4808: // (rom) from JVBandUtils to make it obsolete
4809: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4810: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4811: function CreateIconFromClipboard: TIcon;
4812: { begin JVIconClipboardUtils } { Icon clipboard routines }
4813: function CF_ICON: Word;
4814: procedure AssignClipboardIcon(Icon: TIcon);
4815: { Real-size icons support routines (32-bit only) }
4816: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4817: function CreateRealSizeIcon(Icon: TIcon): HICON;
4818: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4819: {end JVIconClipboardUtils }
4820: function CreateScreenCompatibleDC: HDC;
4821: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF}
4822: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4823: { begin JVRL } // (rom) changed API for inclusion in JCL
4824: procedure RleCompressTo(InStream, OutStream: TStream);
4825: procedure RleDecompressTo(InStream, OutStream: TStream);
4826: procedure RleCompress(Stream: TStream);
4827: procedure RleDecompress(Stream: TStream);
4828: {end JVRL } { begin JVDateUtil }
4829: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4830: function IsLeapYear(AYear: Integer): Boolean;
4831: function DaysInAMonth(const AYear, AMonth: Word): Word;
4832: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4833: function FirstDayOfPrevMonth: TDateTime;
4834: function LastDayOfPrevMonth: TDateTime;
4835: function FirstDayOfNextMonth: TDateTime;
4836: function ExtractDay(ADate: TDateTime): Word;
4837: function ExtractMonth(ADate: TDateTime): Word;
4838: function ExtractYear(ADate: TDateTime): Word;
4839: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4840: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime; inline; {$IFDEF SUPPORTS_INLINE}
4841: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4842: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4843: function ValidDate(ADate: TDateTime): Boolean;
4844: procedure DateDiff(Datet1, Date2: TDateTime; var Days, Months, Years: Word);
4845: function MonthsBetween(Datet1, Date2: TDateTime): Double;
4846: function DaysInPeriod(Datet1, Date2: TDateTime): Longint;
4847: { Count days between Datet1 and Date2 + 1, so if Datet1 = Date2 result = 1 }
4848: function DaysBetween(Datet1, Date2: TDateTime): Longint;
4849: { The same as previous but if Date2 < Datet1 result = 0 }
4850: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4851: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4852: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4853: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4854: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4855: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
4856: { String to date conversions }
4857: function GetDateFormat(const DateFormat: string): TDateOrder;
4858: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4859: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4860: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4861: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4862: //function DefDateFormat(AFourDigitYear: Boolean): string;
4863: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4864: function FormatLongDate(Value: TDateTime): string;
4865: function FormatLongDateTime(Value: TDateTime): string;
4866: { end JVDateUtil }
4867: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4868: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4869: { begin JVStrUtils } { ** Common string handling routines ** }
4870: {$IFDEF UNIX}
4871: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4872: const ToCode, FromCode: AnsiString): Boolean;
4873: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4874: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4875: function OemStrToAnsi(const S: AnsiString): AnsiString;
4876: function AnsiStrToOem(const S: AnsiString): AnsiString;
4877: {$ENDIF UNIX}
4878: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4879: { StrToOem translates a string from the Windows character set into the OEM character set. }
4880: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;

```

```

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

```

```

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

```

```

5057: // restores a window based on it's classname and Caption. Either can be left empty
5058: // to widen the search
5059: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5060: // manipulate the traybar and start button
5061: procedure HideTraybar;
5062: procedure ShowTraybar;
5063: procedure ShowStartButton(Visible: Boolean = True);
5064: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5065: procedure MonitorOn;
5066: procedure MonitorOff;
5067: procedure LowPower;
5068: // send a key to the window named AppName
5069: function SendKey(const AppName: string; Key: Char): Boolean;
5070: {$IFDEF MSWINDOWS}
5071: // returns a list of all win currently visible, the Objects property is filled with their window handle
5072: procedure GetVisibleWindows(List: TStrings);
5073: Function GetVisibleWindowsF( List : TStrings):TStrings';
5074: // associates an extension to a specific program
5075: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5076: procedure AddToRecentDocs(const FileName: string);
5077: function GetRecentDocs: TStringList;
5078: {$ENDIF MSWINDOWS}
5079: function CharIsMoney(const Ch: Char): Boolean;
5080: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5081: function IntToExtended(I: Integer): Extended;
5082: { GetChangedText works out the new text given the current cursor pos & the key pressed
5083:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5084: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5085: function MakeYear4digit(Year, Pivot: Integer): Integer;
5086: //function StrIsInteger(const S: string): Boolean;
5087: function StrIsFloatMoney(const Ps: string): Boolean;
5088: function StrIsDateTime(const Ps: string): Boolean;
5089: function PreformatDateString(Ps: string): string;
5090: function BooleanToInteger(const B: Boolean): Integer;
5091: function StringToBoolean(const Ps: string): Boolean;
5092: function SafeStrToDate(const Ps: string): TDateTime;
5093: function SafeStrToDate(const Ps: string): TDateTime;
5094: function SafeStrToTime(const Ps: string): TDateTime;
5095: function StrDelete(const psSub, psMain: string): string;
5096: { returns the fractional value of pcValue}
5097: function TimeOnly(pcValue: TDateTime): TTime;
5098: { returns the integral value of pcValue }
5099: function DateOnly(pcValue: TDateTime): TDate;
5100: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5101: const { TDateTime value used to signify Null value}
5102: NullEquivalentDate: TDateTime = 0.0;
5103: function DateIsNotNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5104: // Replacement for Win32Check to avoid platform specific warnings in D6
5105: function OSCheck(RetVal: Boolean): Boolean;
5106: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5107:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5108:   not be forced to use FileCtrl unnecessarily }
5109: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5110: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5111: { MinimizeString truncates long string, S, and appends '' symbols, if Length of S is more than MaxLen }
5112: function MinimizeString(const S: string; const MaxLen: Integer): string;
5113: procedure RunDll32Internal(Wnd: THandle; const DLLName, FuncName, CmdLine:string; CmdShow:Integer= SW_SHOWDEFAULT);
5114: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion function and calls it, returning major and
      minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't be
      found.}
5115: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5116: {$ENDIF MSWINDOWS}
5117: procedure ResourceNotFound(ResID: PChar);
5118: function EmptyRect: TRect;
5119: function RectWidth(R: TRect): Integer;
5120: function RectHeight(R: TRect): Integer;
5121: function CompareRect(const R1, R2: TRect): Boolean;
5122: procedure RectNormalize(var R: TRect);
5123: function RectIsSquare(const R: TRect): Boolean;
5124: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5125: //IF AMaxSize = -1 ,then auto calc Square's max size
5126: {$IFDEF MSWINDOWS}
5127: procedure FreeUnusedOle;
5128: function GetWindowsVersion: string;
5129: function LoadDLL(const LibName: string): THandle;
5130: function RegisterServer(const ModuleName: string): Boolean;
5131: function UnregisterServer(const ModuleName: string): Boolean;
5132: {$ENDIF MSWINDOWS}
5133: { String routines }
5134: function GetEnvVar(const VarName: string): string;
5135: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5136: function StringToPChar(var S: string): PChar;
5137: function StrPAalloc(const S: string): PChar;
5138: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5139: function DropT(const S: string): string;
5140: { Memory routines }
5141: function AllocMemo(Size: Longint): Pointer;
5142: function ReallocMeme(fpBlock: Pointer; Size: Longint): Pointer;

```

```

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

```

```

5229: Function IsClassByName( Obj : TObject; ClassName : string ) : Boolean
5230:
5231: procedure SIRegister_MSysUtils(CL: TPSPPascalCompiler);
5232: begin
5233:   Procedure HideTaskBarButton( hWindow : HWND )
5234:   Function msLoadStr( ID : Integer ) : String
5235:   Function msFormat( fmt : String; params : array of const ) : String
5236:   Function msFileExists( const FileName : String ) : Boolean
5237:   Function msIntToStr( Int : Int64 ) : String
5238:   Function msStrPas( const Str : PChar ) : String
5239:   Function msRenameFile( const OldName, NewName : String ) : Boolean
5240:   Function CutFileName( s : String ) : String
5241:   Function GetVersionInfo( var VersionString : String ) : DWORD
5242:   Function FormatTime( t : Cardinal ) : String
5243:   Function msCreateDir( const Dir : string ) : Boolean
5244:   Function SetAutoRun( NeedAutoRun : Boolean; AppName : String ) : Boolean
5245:   Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword ) : DWORD
5246:   Function msStrLen( Str : PChar ) : Integer
5247:   Function msDirectoryExists( const Directory : String ) : Boolean
5248:   Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String ) : String
5249:   Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte ) : LongBool
5250:   Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
5251:   Procedure SetEditWndProc( hWnd : HWND; ptr : TObject )
5252:   Function GetTextFromFile( Filename : String ) : string
5253:   Function IsTopMost( hWnd : HWND ) : Bool // 'LWA_ALPHA', 'LongWord').SetUIInt( $00000002 );
5254:   Function msStrToIntDef( const s : String; const i : Integer ) : Integer
5255:   Function msStrToInt( s : String ) : Integer
5256:   Function GetItemText( hDlg : THandle; ID : DWORD ) : String
5257: end;
5258:
5259: procedure SIRegister_ESBMaths2(CL: TPSPPascalCompiler);
5260: begin
5261:   //TDynFloatArray', 'array of Extended
5262:   TDynLWordArray', 'array of LongWord
5263:   TDynLIntArray', 'array of LongInt
5264:   TDynFloatMatrix', 'array of TDynFloatArray
5265:   TDynLWordMatrix', 'array of TDynLWordArray
5266:   TDynLIntMatrix', 'array of TDynLIntArray
5267:   Function SquareAll( const X : TDynFloatArray ) : TDynFloatArray
5268:   Function InverseAll( const X : TDynFloatArray ) : TDynFloatArray
5269:   Function LnAll( const X : TDynFloatArray ) : TDynFloatArray
5270:   Function Log10All( const X : TDynFloatArray ) : TDynFloatArray
5271:   Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended ) : TDynFloatArray
5272:   Function AddVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5273:   Function SubVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5274:   Function MultVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5275:   Function DotProduct( const X, Y : TDynFloatArray ) : Extended
5276:   Function MNorm( const X : TDynFloatArray ) : Extended
5277:   Function MatrixIsRectangular( const X : TDynFloatMatrix ) : Boolean
5278:   Procedure MatrixDimensions( const X:TDynFloatMatrix; var Rows,Columns:LongWord; var Rectangular:Boolean );
5279:   Function MatrixIsSquare( const X : TDynFloatMatrix ) : Boolean
5280:   Function MatricesSameDimensions( const X, Y : TDynFloatMatrix ) : Boolean
5281:   Function AddMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix
5282:   Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix )
5283:   Function SubtractMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix
5284:   Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix )
5285:   Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended ) : TDynFloatMatrix
5286:   Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended );
5287:   Function MultiplyMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix;
5288:   Function TransposeMatrix( const X : TDynFloatMatrix ) : TDynFloatMatrix;
5289:   Function GrandMean( const X : TDynFloatMatrix; var N : LongWord ) : Extended
5290: end;
5291:
5292: procedure SIRegister_ESBMaths(CL: TPSPPascalCompiler);
5293: begin
5294:   'ESBMinSingle','Single').setExtended( 1.5e-45 );
5295:   'ESBMaxSingle','Single').setExtended( 3.4e+38 );
5296:   'ESBMinDouble','Double').setExtended( 5.0e-324 );
5297:   'ESBMaxDouble','Double').setExtended( 1.7e+308 );
5298:   'ESBMinExtended','Extended').setExtended( 3.6e-4951 );
5299:   'ESBMaxExtended','Extended').setExtended( 1.1e+4932 );
5300:   'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807 );
5301:   'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807 );
5302:   'ESBSqrt2','Extended').setExtended( 1.4142135623730950488 );
5303:   'ESBSqrt3','Extended').setExtended( 1.7320508075688772935 );
5304:   'ESBSqrt5','Extended').setExtended( 2.2360679774997896964 );
5305:   'ESBSqrt10','Extended').setExtended( 3.1622776601683793320 );
5306:   'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729 );
5307:   'ESBCbrt2','Extended').setExtended( 1.2599210498948731648 );
5308:   'ESBCbrt3','Extended').setExtended( 1.4422495703074083823 );
5309:   'ESBCbrt10','Extended').setExtended( 2.1544346900318837219 );
5310:   'ESBCbrt100','Extended').setExtended( 4.641588336127788924 );
5311:   'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630 );
5312:   'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440 );
5313:   'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451 );
5314:   'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928 );
5315:   'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695 );
5316:   'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147 );
5317:   'ESBe','Extended').setExtended( 2.7182818284590452354 );

```

```

5318: 'ESBe2','Extended').setExtended( 7.3890560989306502272);
5319: 'ESBePi','Extended').setExtended( 23.140692632779269006);
5320: 'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5321: 'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5322: 'ESBln2','Extended').setExtended( 0.69314718055994530942);
5323: 'ESBln10','Extended').setExtended( 2.30258509299404568402);
5324: 'ESBlnP1','Extended').setExtended( 1.14472988584940017414);
5325: 'ESBlog10Base2','Extended').setExtended( 3.3219280948873623478);
5326: 'ESBlog2Base10','Extended').setExtended( 0.30102999566398119521);
5327: 'ESBlog3Base10','Extended').setExtended( 0.47712125471966243730);
5328: 'ESBlogPiBase10','Extended').setExtended( 0.4971498726941339);
5329: 'ESBlogEBase10','Extended').setExtended( 0.43429448190325182765);
5330: 'ESBPI','Extended').setExtended( 3.1415926535897932385);
5331: 'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5332: 'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5333: 'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5334: 'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5335: 'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5336: 'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5337: 'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5338: 'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5339: 'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5340: 'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5341: 'ESBTwotoPower63','Extended').setExtended( 9223372036854775808.0);
5342: 'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5343: 'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5344: 'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5345: 'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5346: 'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5347: 'ESBlnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5348: //LongWord', 'Cardinal
5349: TBitList', 'Word
5350: Function UMul( const Num1, Num2 : LongWord) : LongWord
5351: Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5352: Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5353: Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5354: Function SameFloat( const X1, X2 : Extended) : Boolean
5355: Function FloatIsZero( const X : Extended) : Boolean
5356: Function FloatIsPositive( const X : Extended) : Boolean
5357: Function FloatIsNegative( const X : Extended) : Boolean
5358: Procedure IncLim( var B : Byte; const Limit : Byte)
5359: Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5360: Procedure IncLimW( var B : Word; const Limit : Word)
5361: Procedure IncLimI( var B : Integer; const Limit : Integer)
5362: Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5363: Procedure DecLim( var B : Byte; const Limit : Byte)
5364: Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5365: Procedure DecLimW( var B : Word; const Limit : Word)
5366: Procedure DecLimI( var B : Integer; const Limit : Integer)
5367: Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5368: Function MaxB( const B1, B2 : Byte) : Byte
5369: Function MinB( const B1, B2 : Byte) : Byte
5370: Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5371: Function MinSI( const B1, B2 : ShortInt) : ShortInt
5372: Function MaxW( const B1, B2 : Word) : Word
5373: Function MinW( const B1, B2 : Word) : Word
5374: Function esbMaxI( const B1, B2 : Integer) : Integer
5375: Function esbMinI( const B1, B2 : Integer) : Integer
5376: Function MaxL( const B1, B2 : LongInt) : LongInt
5377: Function MinL( const B1, B2 : LongInt) : LongInt
5378: Procedure SwapB( var B1, B2 : Byte)
5379: Procedure SwapSI( var B1, B2 : ShortInt)
5380: Procedure SwapW( var B1, B2 : Word)
5381: Procedure SwapI( var B1, B2 : SmallInt)
5382: Procedure SwapL( var B1, B2 : LongInt)
5383: Procedure SwapI32( var B1, B2 : Integer)
5384: Procedure SwapC( var B1, B2 : LongWord)
5385: Procedure SwapInt64( var X, Y : Int64)
5386: Function esbSign( const B : LongInt) : ShortInt
5387: Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5388: Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5389: Function Max3Word( const X1, X2, X3 : Word) : Word
5390: Function Min3Word( const X1, X2, X3 : Word) : Word
5391: Function MaxBArray( const B : array of Byte) : Byte
5392: Function MaxWArray( const B : array of Word) : Word
5393: Function MaxSIArry( const B : array of ShortInt) : ShortInt
5394: Function MaxIArray( const B : array of Integer) : Integer
5395: Function MaxLArray( const B : array of Longint) : Longint
5396: Function MinBArray( const B : array of Byte) : Byte
5397: Function MinWArray( const B : array of Word) : Word
5398: Function MinSIArry( const B : array of ShortInt) : ShortInt
5399: Function MinIArray( const B : array of Integer) : Integer
5400: Function MinLArray( const B : array of LongInt) : LongInt
5401: Function SumBArray( const B : array of Byte) : Byte
5402: Function SumBArray2( const B : array of Byte) : Word
5403: Function SumSIArry( const B : array of ShortInt) : ShortInt
5404: Function SumSIArry2( const B : array of ShortInt) : Integer
5405: Function SumWArray( const B : array of Word) : Word
5406: Function SumWArray2( const B : array of Word) : LongInt

```

```

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

```

```

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

```

```

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

```

```

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

```

```

5745: Function CopyLeftS( const S : ShortString; Len : Cardinal ) : ShortString
5746: Function CopyMidS( const S : ShortString; First, Len : Cardinal ) : ShortString
5747: Function CopyRightS( const S : ShortString; First : Cardinal ) : ShortString
5748: Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal ) : ShortString
5749: Function CopyFromNthWordS( const S, WordDelims : string; const AWord : String; N : Cardinal; var
  SubString : ShortString ) : Bool;
5750: Function DeleteFromNthWordS( const S, WordDelims : String; AWord : ShortString; N : Cardinal; var
  SubStr : ShortString ) : Bool;
5751: Function CopyFromToWordS( const S, WordDelims, Word1, Word2 : ShortString; N1, N2 : Cardinal; var
  SubString : ShortString ) : Bool;
5752: Function DeleteFromToWords( const S, WordDelims, Wrld1, Wrld2 : ShortString; N1, N2 : Cardinal; var
  SubString : ShortString ) : Bool;
5753: Function CopyWithins( const S, Delimiter : ShortString; Strip : Boolean ) : ShortString
5754: Function DeleteWithinS( const S, Delimiter : ShortString ) : ShortString
5755: Function ExtractTokensS( const S,
  Delims : ShortString; QuoteChar : AnsiChar; AllowNulls : Boolean; Tokens : TStrings ) : Cardinal
5756: Function IsChAlphaS( C : Char ) : Boolean
5757: Function IsChNumericS( C : Char; const Numbers : ShortString ) : Boolean
5758: Function IsChAlphaNumericS( C : Char; const Numbers : ShortString ) : Boolean
5759: Function IsStrAlphaS( const S : ShortString ) : Boolean
5760: Function IsStrNumericS( const S, Numbers : ShortString ) : Boolean
5761: Function IsStrAlphaNumericS( const S, Numbers : ShortString ) : Boolean
5762: Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal ) : Boolean
5763: Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal ) : Boolean
5764: Function LastStringS( const S, AString : ShortString; var Position : Cardinal ) : Boolean
5765: Function LeftTrimCharsS( const S, Chars : ShortString ) : ShortString
5766: Function KeepCharsS( const S, Chars : ShortString ) : ShortString
5767: Function RepeatStringS( const RepeatString : ShortString; var Repetitions : Cardinal; MaxLen : Cardinal ) : ShortString;
5768: Function ReplaceStringS( const S, OldStr, NewStr : ShortString; N : Cardinal; var
  Replacements : Cardinal ) : ShortString;
5769: Function ReplaceStringAllS( const S, OldString, NewString : ShortString; var Replacements : Cardinal ) : ShortString;
5770: Function ReplaceWordS( const S, WordDelims, OldWord, NewW : String; N : Cardinal; var
  Replacements : Cardinal ) : ShortString;
5771: Function ReplaceWordAllS( const S, WordDelims, OldWord, NewWord : ShortString; var
  Replacements : Cardinal ) : ShortString;
5772: Function RightTrimCharsS( const S, Chars : ShortString ) : ShortString
5773: Function StrWithinS( const S, SearchStr : ShortString; Start : Cardinal; var Position : Cardinal ) : boolean
5774: Function TrimCharsS( const S, Chars : ShortString ) : ShortString
5775: Function WordPosS( const S, WordDelims, AWord : ShortString; N : Cardinal; var Position : Cardinal ) : Boolean
5776: end;
5777:
5778:
5779: *****unit uPSI_StUtils; from SysTools4*****
5780: Function SignL( L : Longint ) : Integer
5781: Function SignF( F : Extended ) : Integer
5782: Function MinWord( A, B : Word ) : Word
5783: Function MidWord( W1, W2, W3 : Word ) : Word
5784: Function MaxWord( A, B : Word ) : Word
5785: Function MinLong( A, B : LongInt ) : LongInt
5786: Function MidLong( L1, L2, L3 : LongInt ) : LongInt
5787: Function MaxLong( A, B : LongInt ) : LongInt
5788: Function MinFloat( F1, F2 : Extended ) : Extended
5789: Function MidFloat( F1, F2, F3 : Extended ) : Extended
5790: Function MaxFloat( F1, F2 : Extended ) : Extended
5791: Function MakeInteger16( H, L : Byte ) : SmallInt
5792: Function MakeWordS( H, L : Byte ) : Word
5793: Function SwapNibble( B : Byte ) : Byte
5794: Function SwapWord( L : LongInt ) : LongInt
5795: Procedure SetFlag( var Flags : Word; FlagMask : Word )
5796: Procedure ClearFlag( var Flags : Word; FlagMask : Word )
5797: Function FlagIsSet( Flags, FlagMask : Word ) : Boolean
5798: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte )
5799: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte )
5800: Function ByteFlagIsSet( Flags, FlagMask : Byte ) : Boolean
5801: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt )
5802: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt )
5803: Function LongFlagIsSet( Flags, FlagMask : LongInt ) : Boolean
5804: Procedure ExchangeBytes( var I, J : Byte )
5805: Procedure ExchangeWords( var I, J : Word )
5806: Procedure ExchangeLongInts( var I, J : LongInt )
5807: Procedure ExchangeStructs( var I, J, Size : Cardinal )
5808: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word )
5809: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal )
5810: Function AddWordToPtr( P : __Pointer; W : Word ) : __Pointer
5811: //*****uPSI_StFIN;*****
5812: Function AccruedInterestMaturity( Issue, Maturity : TStDate; Rate, Par : Extended; Basis : TStBasis ) : Extended
5813: Function AccruedInterestPeriodic( Issue, Settlement, Maturity : TStDate; Rate,
  Par : Extended; Frequency : TStFrequency; Basis : TStBasis ) : Extended
5814: Function BondDuration( Settlement, Maturity : TStDate; Rate,
  Yield : Ext; Frequency : TStFrequency; Basis : TStBasis ) : Extended;
5815: Function BondPrice( Settlement, Maturity : TStDate; Rate, Yield, Redempt : Ext; Freq : TStFrequency; Basis : TStBasis ) :
  Extended
5816: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod : Integer;
  Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5817: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod : Integer;
  Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5818: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis ) : LongInt
5819: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer ) : Extended

```

```

5820: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5821: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer ) : Extended
5822: Function DollarToDecimalText( DecDollar : Extended ) : string
5823: Function DollarToFraction( DecDollar : Extended; Fraction : Integer ) : Extended
5824: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer ) : string
5825: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency ) : Extended
5826: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
    PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5827: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double ) : Extended
5828: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer ) : Extended
5829: Function InterestRateS(NPeriods:Int;Pmt,PV,
    FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5830: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended ) : Extended
5831: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended ) : Extended
5832: Function IsCardValid( const S : string ) : Boolean
5833: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
    Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5834: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5835: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5836: Function NetPresentValueS( Rate : Extended; const Values : array of Double ) : Extended
5837: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer ) : Extended
5838: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency ) : Extended
5839: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5840: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5841: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
    : TStPaymentTime ) : Extended
5842: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5843: Function PresentValueS( Rate : Extended; NPeriods : Integer; Pmt, FV : Extended; Frequency : TStFrequency;
    Timing : TStPaymentTime ) : Extended
5844: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5845: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean ) : Extended
5846: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5847: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5848: Function TBillyYield( Settlement, Maturity : TStDate; Price : Extended ) : Extended
5849: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
    Factor : Extended; NoSwitch : boolean ) : Extended
5850: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5851: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
    Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5852: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5853:
5854: //*****unit uPSI_StAstroP;
5855: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray )
5856: //*****unit uPSI_StStat; Statistic Package of SysTools*****
5857: Function AveDev( const Data : array of Double ) : Double
5858: Function AveDev16( const Data, NData : Integer ) : Double
5859: Function Confidence( Alpha, StandardDev : Double; Size : LongInt ) : Double
5860: Function Correlation( const Data1, Data2 : array of Double ) : Double
5861: Function Correlation16( const Data1, Data2, NData : Integer ) : Double
5862: Function Covariance( const Data1, Data2 : array of Double ) : Double
5863: Function Covariance16( const Data1, Data2, NData : Integer ) : Double
5864: Function DevSq( const Data : array of Double ) : Double
5865: Function DevSq16( const Data, NData : Integer ) : Double
5866: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5867: //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5868: Function GeometricMeanS( const Data : array of Double ) : Double
5869: Function GeometricMean16( const Data, NData : Integer ) : Double
5870: Function HarmonicMeanS( const Data : array of Double ) : Double
5871: Function HarmonicMean16( const Data, NData : Integer ) : Double
5872: Function Largest( const Data : array of Double; K : Integer ) : Double
5873: Function Largest16( const Data, NData : Integer; K : Integer ) : Double
5874: Function MedianS( const Data : array of Double ) : Double
5875: Function Median16( const Data, NData : Integer ) : Double
5876: Function Mode( const Data : array of Double ) : Double
5877: Function Model6( const Data, NData : Integer ) : Double
5878: Function Percentile( const Data : array of Double; K : Double ) : Double
5879: Function Percentile16( const Data, NData : Integer; K : Double ) : Double
5880: Function PercentRank( const Data : array of Double; X : Double ) : Double
5881: Function PercentRank16( const Data, NData : Integer; X : Double ) : Double
5882: Function Permutations( Number, NumberChosen : Integer ) : Extended
5883: Function Combinations( Number, NumberChosen : Integer ) : Extended
5884: Function Factorials( N : Integer ) : Extended
5885: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean ) : Integer
5886: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean ) : Integer
5887: Function Smallest( const Data : array of Double; K : Integer ) : Double
5888: Function Smallest16( const Data, NData : Integer; K : Integer ) : Double
5889: Function TrimMean( const Data : array of Double; Percent : Double ) : Double
5890: Function TrimMean16( const Data, NData : Integer; Percent : Double ) : Double
5891: AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5892:     +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5893: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
    LF:TStLinEst;ErrorStats:Bool;
5894: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
    LF:TStLinEst;ErrorStats:Bool;
5895: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5896: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5897: Function Intercept( const KnownY : array of Double; const KnownX : array of Double ) : Double
5898: Function RSquared( const KnownY : array of Double; const KnownX : array of Double ) : Double
5899: Function Slope( const KnownY : array of Double; const KnownX : array of Double ) : Double

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

6427: ASSOCSTR_FRIENDLYAPPNAME', 'LongInt'( 4);
6428: ASSOCSTR_NOOPEN', 'LongInt'( 5);
6429: ASSOCSTR_SHELLNEWVALUE', 'LongInt'( 6);
6430: ASSOCSTR_DDECOMMAND', 'LongInt'( 7);
6431: ASSOCSTR_DDEIFEXEC', 'LongInt'( 8);
6432: ASSOCSTR_DDEAPPLICATION', 'LongInt'( 9);
6433: ASSOCSTR_DDETOPIC', 'LongInt'( 10);
6434: ASSOCSTR_INFOTIP', 'LongInt'( 11);
6435: ASSOCSTR_MAX', 'LongInt'( 12);
6436: ASSOCSTR', 'DWORD
6437: ASSOCKEY_SHELLEXECCLASS', 'LongInt'( 1);
6438: ASSOCKEY_APP', 'LongInt'( 2);
6439: ASSOCKEY_CLASS', 'LongInt'( 3);
6440: ASSOCKEY_BASECLASS', 'LongInt'( 4);
6441: ASSOCKEY_MAX', 'LongInt'( 5);
6442: ASSOCKEY', 'DWORD
6443: ASSOCDATA_MSIDSCRIPTOR', 'LongInt'( 1);
6444: ASSOCDATA_NOACTIVATEHANDLER', 'LongInt'( 2);
6445: ASSOCDATA_QUERYCLASSTORE', 'LongInt'( 3);
6446: ASSOCDATA_HASPERUSERASSOC', 'LongInt'( 4);
6447: ASSOCDATA_MAX', 'LongInt'( 5);
6448: ASSOCDATA', 'DWORD
6449: ASSOCENUM_NONE', 'LongInt'( 0);
6450: ASSOCENUM', 'DWORD
6451: SID_IQueryAssociations', 'String' '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6452: SHACF_DEFAULT $00000000;
6453: SHACF_FILESYSTEM', 'LongWord( $00000001);
6454: SHACF_URLHISTORY', 'LongWord( $00000002);
6455: SHACF_URLMRU', 'LongWord( $00000004);
6456: SHACF_USETAB', 'LongWord( $00000008);
6457: SHACF_FILESYS_ONLY', 'LongWord( $00000010);
6458: SHACF_AUTOSUGGEST_FORCE_ON', 'LongWord( $10000000);
6459: SHACF_AUTOSUGGEST_FORCE_OFF', 'LongWord( $20000000);
6460: SHACF_AUTOAPPEND_FORCE_ON', 'LongWord( $40000000);
6461: SHACF_AUTOAPPEND_FORCE_OFF', 'LongWord( DWORD( $80000000 ) );
6462: Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD ) : HRESULT
6463: Procedure SHSetThreadRef( punk : IUnknown )
6464: Procedure SHGetThreadRef( out ppunk : IUnknown )
6465: CTF_INSIST', 'LongWord( $00000001);
6466: CTF_THREAD_REF', 'LongWord( $00000002);
6467: CTF_PROCESS_REF', 'LongWord( $00000004);
6468: CTF_COINIT', 'LongWord( $00000008);
6469: Function SHCreateShellPalette( hdc : HDC ) : HPALETTE
6470: Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD )
6471: Function ColorHLSTORGB( whue, wLuminance, wSaturation : WORD ) : TColorRef
6472: Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean ) : TColorRef
6473: Function GetSysColorBrush( nIndex : Integer ) : HBRUSH
6474: Function DrawFocusRect( hdc : HDC; const lprc : TRect ) : BOOL
6475: Function FillRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH ) : Integer
6476: Function FrameRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH ) : Integer
6477: Function InvertRect( hdc : HDC; const lprc : TRect ) : BOOL
6478: Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer ) : BOOL
6479: Function SetRectEmpty( var lprc : TRect ) : BOOL
6480: Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect ) : BOOL
6481: Function InflateRect( var lprc : TRect; dx, dy : Integer ) : BOOL
6482: Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect ) : BOOL
6483: Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect ) : BOOL
6484:
6485: Function InitializeFlatSB( hWnd : HWND ) : Bool
6486: Procedure UninitializeFlatSB( hWnd : HWND )
6487: Function FlatSB_GetScrollProp( pl : HWND; propIndex : Integer; p3 : PInteger ) : Bool
6488: Function FlatSB_SetScrollProp( pl : HWND; index : Integer; newValue : Integer; p4 : Bool ) : Bool
6489: Function GET_APCOMMAND_LPARAM( lParam : Integer ) : Word //of JvWin32
6490: Function GET_DEVICE_LPARAM( lParam : Integer ) : Word
6491: Function GET_MOUSEORKEY_LPARAM( lParam : Integer ) : Integer
6492: Function GET_FLAGS_LPARAM( lParam : Integer ) : Word
6493: Function GET_KEYSTATE_LPARAM( lParam : Integer ) : Word
6494:
6495:
6496: // **** 204 unit uPSI_ShellAPI;
6497: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT ) : UINT
6498: Function DragQueryPoint( Drop : HDROP; var Point : TPoint ) : BOOL
6499: Procedure DragFinish( Drop : HDROP )
6500: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL )
6501: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar,ShowCmd:Integer):HINST
6502: Function FindExecutable( FileName, Directory : PChar; Result : PChar ) : HINST
6503: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON ) : Integer
6504: Function DuplicateIcon( hInst : HINST; Icon : HICON ) : HICON
6505: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word ) : HICON
6506: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT ) : HICON
6507: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData ) : UINT
6508: Function DoEnvironmentSubst( szString : PChar; cbString : UINT ) : DWORD
6509: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6510: Procedure SHFreeNameMappings( hNameMappings : THandle )
6511:
6512: DLLVER_PLATFORM_WINDOWS', 'LongWord( $00000001);
6513: DLLVER_PLATFORM_NT', 'LongWord( $00000002);
6514: DLLVER_MAJOR_MASK', 'LongWord( Int64( $FFFF000000000000 ) );
6515: DLLVER_MINOR_MASK', 'LongWord( Int64( $0000FFFF00000000 ) );

```

```

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

```

```

6588: Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings ) : Boolean
6589: Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
NamesList:TStrings):Bool
6590: Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings ) : Boolean
6591: Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
Descript:Bool):Bool;
6592: Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings ) : Boolean;
6593: Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings ) : Boolean;
6594: Function PeCreateRequiredImportList(const FileName : TFileName; RequiredImportsList: TStrings): Boolean;
6595: //Function PeMapImgNtHeaders( const BaseAddress : Pointer ) : PImageNtHeaders
6596: //Function PeMapImgLibraryName( const BaseAddress : Pointer ) : string
6597: //Function PeMapImgSections( const NtHeaders : PImageNtHeaders ) : PImageSectionHeader
6598: //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string ) :
PImageSectionHeader
6599: //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6600: //Function PeMapImgResolvePackageThunk( Address : Pointer ) : Pointer
6601: Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
__Pointer;
6602: SIRegister_TJclPeSectionStream(CL);
6603: SIRegister_TJclPeMapImgHookItem(CL);
6604: SIRegister_TJclPeMapImgHooks(CL);
6605: //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer,var
NtHeaders:TImageNtHeaders):Boolean
6606: //Function PeDdbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6607: Type TJclBorUmSymbolKind', '(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6608: TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6609: TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6610: TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6611: TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6612: TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6613: Function PeBorUnmangleName( const Name : string; var Ummangled : string; var Description :
TJclBorUmDescription; var BasePos : Integer ) : TJclBorUmResult;
6614: Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
Description:TJclBorUmDescription):TJclBorUmResult;
6615: Function PeBorUnmangleName2( const Name : string; var Unmangled : string ) : TJclBorUmResult;
6616: Function PeBorUnmangleName3( const Name : string ) : string;
6617: Function PeIsNameMangled( const Name : string ) : TJclPeUmResult
6618: Function PeUnmangleName( const Name : string; var Unmangled : string ) : TJclPeUmResult
6619:
6620:
6621: //***** SysTools uPSI_StSystem; ****
6622: Function StCopyFile( const SrcPath, DestPath : AnsiString ) : Cardinal
6623: Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString ) : AnsiString
6624: Function DeleteVolumeLabel( Drive : Char ) : Cardinal
6625: //Procedure EnumerateDirectories(const
StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6626: //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
IncludeItem:TIncludeItemFunc);
6627: Function FileHandlesLeft( MaxHandles : Cardinal ) : Cardinal
6628: Function FileMatchesMask( const FileName, FileMask : AnsiString ) : Boolean
6629: Function FileTimeToStDateTime( FileTime : LongInt ) : TStDateTimeRec
6630: Function FindNthSlash( const Path : AnsiString; n : Integer ) : Integer
6631: Function FlushOsBuffers( Handle : Integer ) : Boolean
6632: Function GetCurrentUser : AnsiString
6633: Function GetDiskClass( Drive : Char ) : DiskClass
6634: Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
SectorsPerCluster:Cardinal):Bool;
6635: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
DiskSize:Double):Bool;
6636: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
DiskSize:Comp):Boolean;
6637: { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes }
6638: Function GetDiskSpace2(const path: String; index: integer): int64;
6639: Function GetFileCreateDate( const FileName : AnsiString ) : TDateTime
6640: Function StGetFileLastAccess( const FileName : AnsiString ) : TDateTime
6641: Function GetFileLastModify( const FileName : AnsiString ) : TDateTime
6642: Function GetHomeFolder( aForceSlash : Boolean ) : AnsiString
6643: Function GetLongPath( const APath : AnsiString ) : AnsiString
6644: Function GetMachineName : AnsiString
6645: Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType ) : Cardinal
6646: Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean ) : AnsiString
6647: Function GetShortPath( const APath : AnsiString ) : AnsiString
6648: Function GetSystemFolder( aForceSlash : Boolean ) : AnsiString
6649: Function GetTempFolder( aForceSlash : boolean ) : AnsiString
6650: Function StGetWindowsFolder( aForceSlash : boolean ) : AnsiString
6651: Function GetWorkingFolder( aForceSlash : boolean ) : AnsiString
6652: Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer ) : TStDateTimeRec
6653: Function StIsDirectory( const DirName : AnsiString ) : Boolean
6654: Function IsDirectoryEmpty( const S : AnsiString ) : Integer
6655: Function IsDriveReady( Drive : Char ) : Boolean
6656: Function IsFile( const FileName : AnsiString ) : Boolean
6657: Function IsFileArchive( const S : AnsiString ) : Integer
6658: Function IsFileHidden( const S : AnsiString ) : Integer
6659: Function IsFileReadOnly( const S : AnsiString ) : Integer
6660: Function IsFileSystem( const S : AnsiString ) : Integer
6661: Function LocalDateTimeToGlobal( const DTL : TStDateTimeRec; MinOffset : Integer ) : TStDateTimeRec
6662: Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char ) : Cardinal
6663: Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer ) : Boolean
6664: Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType ) : Cardinal

```

```

6665: Procedure SplitPath( const APath : AnsiString; Parts : TStrings )
6666: Function StDateTimeToFileTime( const FileTime : TStDateTimeRec ) : LongInt
6667: Function StDateTimeToUnixTime( const DTL : TStDateTimeRec ) : Longint
6668: Function UnixTimeToStDateTime( UnixTime : Longint ) : TStDateTimeRec
6669: Function ValidDrive( Drive : Char ) : Boolean
6670: Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char ) : Cardinal
6671:
6672: //*****unit uPST_Jc1LANMan;*****
6673: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
  const PasswordNeverExpires : Boolean ) : Boolean
6674: Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
  const PasswordNeverExpires : Boolean ) : Boolean
6675: Function DeleteAccount( const Servername, Username : string ) : Boolean
6676: Function DeleteLocalAccount( Username : string ) : Boolean
6677: Function CreateLocalGroup( const Server, Groupname, Description : string ) : Boolean
6678: Function CreateGlobalGroup( const Server, Groupname, Description : string ) : Boolean
6679: Function DeleteLocalGroup( const Server, Groupname : string ) : Boolean
6680: Function GetLocalGroups( const Server : string; const Groups : TStrings ) : Boolean
6681: Function GetGlobalGroups( const Server : string; const Groups : TStrings ) : Boolean
6682: Function LocalGroupExists( const Group : string ) : Boolean
6683: Function GlobalGroupExists( const Server, Group : string ) : Boolean
6684: Function AddAccountToLocalGroup( const Accountname, Groupname : string ) : Boolean
6685: Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID ) : string
6686: Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string )
6687: Function IsLocalAccount( const AccountName : string ) : Boolean
6688: Function TimeStampInterval( StartStamp, EndStamp : TDateTime ) : integer
6689: Function GetRandomString( NumChar : cardinal ) : string
6690:
6691: //*****unit uPST_cUtils;*****
6692: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )'
6693: Function cIsWinNT : boolean
6694: Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
  Multitasking:Boolean;
6695: Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ) : THandle
6696: Function cRunAndGetOutput( Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
  CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean ) : string
6697: Function cGetShortName( FileName : string ) : string
6698: Procedure cShowError( Msg : String )
6699: Function cCommaStrToStr( s : string; formatstr : string ) : string
6700: Function cIncludeQuoteIfSpaces( s : string ) : string
6701: Function cIncludeQuoteIfNeeded( s : string ) : string
6702: Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream )
6703: Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6704: Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET ) : boolean;
6705: Function cBuildFilter1( var value : string; const _filters : array of string ) : boolean;
6706: Function cCodeInstoStr( s : string ) : string
6707: Function cStrtoCodeIns( s : string ) : string
6708: Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6709: Function cAttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6710: Procedure cStrtoPoint( var pt : TPoint; value : string )
6711: Function cPointtoStr( const pt : TPoint ) : string
6712: Function cListtoStr( const List : TStrings ) : string
6713: Function ListtoStr( const List : TStrings ) : string
6714: Procedure StrtoList( s : string; const List : TStrings; const delimiter : char )
6715: Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char )
6716: Function cGetFileType( const FileName : string ) : TUnitType
6717: Function cGetExTyp( const FileName : string ) : TExUnitType
6718: Procedure cSetPath( Add : string; const UseOriginal : boolean )
6719: Function cExpandFileto( const FileName : string; const basePath : string ) : string
6720: Function cFileSamePath( const FileName : string; const TestPath : string ) : boolean
6721: Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem )
6722: Function cGetLastPos( const SubStr : string; const S : string ) : integer
6723: Function cGenMakePath( FileName : String ) : String;
6724: Function cGenMakePath2( FileName : String ) : String
6725: Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean ) : String;
6726: Function cGetRealPath( BrokenFileName : String; Directory : String ) : String
6727: Function cCalcMod( Count : Integer ) : Integer
6728: Function cGetVersionString( FileName : string ) : string
6729: Function cCheckChangeDir( var Dir : string ) : boolean
6730: Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6731: Function cIsNumeric( s : string ) : boolean
6732: Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6733: Function AttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6734: Function GetFileType( const FileName : string ) : TUnitType
6735: Function Atoi(const aStr: string): integer
6736: Function Itoa(const aint: integer): string
6737: Function Atof(const aStr: string): double';
6738: Function Atol(const aStr: string): longint';
6739:
6740:
6741: procedure SIRegister_cHTTP_Utils(CL: TPPascalCompiler);
6742: begin
6743:   FindClass( 'TOBJECT' ), 'EHTTP
6744:   FindClass( 'TOBJECT' ), 'EHTTPParser
6745:   //AnsiCharSet', 'set of AnsiChar
6746:   AnsiStringArray', 'array of AnsiString
6747:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6748:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6749:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
```

```

6750: +'TPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6751: +'CustomMinVersion : Integer; end
6752: THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6753: +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6754: +'anguage, hntContentEncoding, hntTransferEncoding, hntServer, hntU'
6755: +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6756: +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6757: +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6758: +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges, '
6759: +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSince'
6760: +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6761: +'nection, hntOrigin, hntKeepAlive )'
6762: THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6763: THTTPCustomHeader', 'record FieldName : AnsiString;FieldValue :'
6764: +' AnsiString; end
6765: //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6766: THTTPContentLengthEnum', '( hcNone, hc1tByteCount )'
6767: THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6768: //PHTTPCContentLength', '^THTTPCContentLength // will not work
6769: THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )'
6770: THTTPContentTypeEnum', '( hcNone, hctCustomParts, hctCustomStri'
6771: +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6772: +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6773: +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctAppli'
6774: +'ctionCustom, hctAudioCustom, hctVideoCustom )'
6775: THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6776: +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray; '
6777: +'CustomStr : AnsiString; end
6778: THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )'
6779: THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6780: +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6781: +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6782: +'String; DateTime : TDateTime; Custom : AnsiString; end
6783: THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )'
6784: THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnum'
6785: +'m; Custom : AnsiString; end
6786: THTTPConnectionFieldEnum', '( hcfcNone, hcfcCustom, hcfcClose, hcfcKeepAlive )'
6787: THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum; '
6788: +' Custom : AnsiString; end
6789: THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )'
6790: THTTPAgeField', 'record Value : THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6791: THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )'
6792: THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6793: +' , hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )'
6794: THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6795: +' , hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6796: +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )'
6797: THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end'
6798: THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6799: +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )'
6800: THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end'
6801: THTTPContentEncodingFieldEnum', '( hcelfNone, hcelfList )'
6802: THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6803: +'FieldEnum; List : array of THTTPContentEncoding; end
6804: THTTPRetryAfterFieldEnum', '( hrarfNone, hrarfCustom, harfDate, harfSeconds )'
6805: THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum; '
6806: +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6807: THTTPContentRangeFieldEnum', '( hcrlenNone, hcrlenCustom, hcrlenByteRange )'
6808: THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6809: +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6810: THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )'
6811: THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6812: THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField'
6813: THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6814: +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6815: +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6816: +'CustomFieldArray; Custom : AnsiString; end
6817: //PHTTPSSetCookieField', '^THTTPSSetCookieField // will not work
6818: THTTPSSetCookieFieldArray', 'array of THTTPSSetCookieField
6819: THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )'
6820: THTTPCookiefieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6821: //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6822: THTTPCookiefieldEntryArray', 'array of THTTPCookieFieldEntry
6823: THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6824: +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6825: THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6826: +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength; '
6827: +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6828: +' ; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6829: THTTPCustomHeaders', 'array of THTTPCustomHeader
6830: //THTTFFixedHeaders', 'array[THTTPHeaderNameEnum] of AnsiString
6831: THTTFFixedHeaders', 'array[0..42] of AnsiString
6832: THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6833: +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )'
6834: THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6835: THTTPRequestStartLine', 'record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6836: THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders; '
6837: +' FixedHeaders : THTTFFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6838: +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end

```

```

6839: //^THTTTPRequestHeader', '^THTTTPRequestHeader // will not work
6840: THTTTPRequest', 'record StartLine : THTTTPRequestStartLine; Header'
6841: +' : THTTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6842: THTTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6843: THTTTPResponseStartLine', 'record Version : THTTTPVersion; Code : '
6844: +'Integer; Msg : THTTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6845: THTTTPResponseHeader', 'record CommonHeaders : THTTTPCommonHeaders'
6846: +' ; FixedHeaders : THTTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6847: +'okies : THTTPSSetCookieFieldArray; Expires : THTTTPDateField; LastModified : '
6848: +' THTTTPDateField; Age : THTTPAgeField; end
6849: //^THTTTPResponseHeader', '^THTTTPResponseHeader // will not work
6850: THTTTPResponse', 'record StartLine : THTTTPResponseStartLine; Head'
6851: +'er : THTTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6852: Function HTTPMessageHasContent( const H : THTTTPCommonHeaders ) : Boolean
6853: Procedure InitTHTTTPRequest( var A : THTTTPRequest )
6854: Procedure InitTHTTTPResponse( var A : THTTTPResponse )
6855: Procedure ClearTHTTTPVersion( var A : THTTTPVersion )
6856: Procedure ClearTHTTTPContentLength( var A : THTTTPContentLength )
6857: Procedure ClearTHTTTPContentType( var A : THTTTPContentType )
6858: Procedure ClearTHTTTPDateField( var A : THTTTPDateField )
6859: Procedure ClearTHTTTPTransferEncoding( var A : THTTTPTransferEncoding )
6860: Procedure ClearTHTTTPConnectionField( var A : THTTTPConnectionField )
6861: Procedure ClearTHTTTPAgeField( var A : THTTPAgeField )
6862: Procedure ClearTHTTTPContentEncoding( var A : THTTTPContentEncoding )
6863: Procedure ClearTHTTTPContentEncodingField( var A : THTTTPContentEncodingField )
6864: Procedure ClearTHTTTPContentRangeField( var A : THTTTPContentRangeField )
6865: Procedure ClearTHTTPSSetCookieField( var A : THTTPSSetCookieField )
6866: Procedure ClearTHTTTPCommonHeaders( var A : THTTTPCommonHeaders )
6867: //Procedure ClearTHTTTPFixedHeaders( var A : THTTTPFixedHeaders )
6868: Procedure ClearTHTTTPCustomHeaders( var A : THTTPCustomHeaders )
6869: Procedure ClearTHTTPCookieField( var A : THTTPCookieField )
6870: Procedure ClearTHTTTPMethod( var A : THTTTPMethod )
6871: Procedure ClearTHTTTPRequestStartLine( var A : THTTTPRequestStartLine )
6872: Procedure ClearTHTTTPRequestHeader( var A : THTTTPRequestHeader )
6873: Procedure ClearTHTTTPRequest( var A : THTTTPRequest )
6874: Procedure ClearTHTTTPResponseStartLine( var A : THTTTPResponseStartLine )
6875: Procedure ClearTHTTTPResponseHeader( var A : THTTTPResponseHeader )
6876: Procedure ClearTHTTTPResponse( var A : THTTTPResponse )
6877: THTTTPStringOption', '( hsoNone )
6878: THTTTPStringOptions', 'set of THTTTPStringOption
6879: FindClass('TOBJECT'), 'TAnsiStringBuilder
6880:
6881: Procedure BuildStrTHTTTPVersion(const A:THTTTPVersion;const B:TAnsiStringBuilder; P:THTTTPStringOptions;
6882: Procedure BuildStrTHTTTPContentLengthValue( const
6883: A:THTTTPContentLength;B:TAnsiStringBuilder;P:THTTTPStringOptions )
6884: Procedure BuildStrTHTTTPContentLength( const A : THTTTPContentLength;
6885: B:TAnsiStringBuilder;P:THTTTPStringOptions )
6886: Procedure BuildStrTHTTTPContentTypeValue( const A : THTTTPContentType;B:TAnsiStringBuilder;const
6887: P:THTTTPStringOptions )
6888: Procedure BuildStrTHTTTPContentType( const A : THTTTPContType;const B:TAnsiStringBuilder; const
6889: P:THTTTPStringOptions )
6890: Procedure BuildStrTHTTTPRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
6891: B : TAnsiStringBuilder; const P : THTTTPStringOptions )
6892: Procedure BuildStrTHTTTPDateFieldValue( const A : THTTTPDateField; const B : TAnsiStringBuilder; const P :
6893: THTTTPStringOptions )
6894: Procedure BuildStrTHTTTPDateField( const A : THTTTPDateField;const B:TAnsiStringBuilder;const
6895: P:THTTTPStringOptions );
6896: Procedure BuildStrTHTTTPTransferEncodingValue( const A : THTTTPTransferEncoding; const B :
6897: TAnsiStringBuilder; const P : THTTTPStringOptions )
6898: Procedure BuildStrTHTTTPTransferEncoding( const A : THTTTPTransferEncoding; const B : TAnsiStringBuilder;
6899: const P : THTTTPStringOptions )
6900: Procedure BuildStrTHTTTPContentEncoding( const A : THTTTPContentEncoding; const B : TAnsiStringBuilder;
6901: const P : THTTTPStringOptions )
6902: Procedure BuildStrTHTTTPContentEncodingField( const A:THTTTPContentEncodingField;const
6903: B:TAnsiStringBuilder;const P:THTTTPStringOptions )
6904: Procedure BuildStrTHTTTPProxyConnectionField( const A : THTTTPConnectionField; const B : TAnsiStringBuilder;
6905: const P : THTTTPStringOptions )
6906: Procedure BuildStrTHTTTPCommonHeaders( const A : THTTTPCommonHeaders; const B : TAnsiStringBuilder; const P
6907: : THTTTPStringOptions )
6908: Procedure BuildStrTHTTTPFixedHeaders( const A:THTTTPFixedHeaders;const B:TAnsiStrBuilder,const
6909: P:THTTTPStringOptions )
6910: Procedure BuildStrTHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
6911: : THTTTPStringOptions )
6912: Procedure BuildStrTHTTPSSetCookieFieldValue( const A : THTTPSSetCookieField; const B : TAnsiStringBuilder;
6913: const P : THTTTPStringOptions )
6914: Procedure BuildStrTHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const
6915: P:THTTTPStringOptions );
6916: Procedure BuildStrTHTTTPMethod( const A : THTTTPMethod; const B : TAnsiStringBuilder; const P :
6917: THTTTPStringOptions )

```

```

6905: Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
6906: const P : THTTPStringOptions)
6907: Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
6908: P:THTTPStringOptions);
6909: Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
6910: THTTPStringOptions)
6911: Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
6912: TAnsiStringBuilder; const P : THTTPStringOptions)
6913: Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
6914: THTTPStrOptions);
6915: Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
6916: P:THTTPStringOptions);
6917: Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
6918: P:THTTPStringOptions);
6919: Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6920: SIRRegister_THTPParser(CL);
6921: FindClass ('TOBJECT'), 'THTTPContentDecoder'
6922: THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6923: THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6924: THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,
6925: +' crcsContentCRLF, crcsTrailer, crcsFinished )
6926: THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6927: SIRRegister_THTPPContentDecoder(CL);
6928: THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6929: FindClass ('TOBJECT'), 'THTTPContentReader
6930: THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6931: THTTPContentReaderLogEvent', 'Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
6932: LogLevel:Int;
6933: SIRRegister_THTPPContentReader(CL);
6934: THTTPContentWriterMechanism', '( hctmEvent, hctmString, hctmStream, hctmFile )
6935: FindClass ('TOBJECT'), 'THTTPContentWriter
6936: THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6937: SIRRegister_THTPPContentWriter(CL);
6938: Procedure SelfTestcHTTPUtils
6939: end;
6940: (*-----*)
6941: procedure SIRRegister_cTLSUtils(CL: TPPascalCompiler);
6942: begin
6943: 'TLSLibraryVersion', 'String '1.00
6944: 'TLSerror_None', 'LongInt'( 0 );
6945: 'TLSerror_InvalidBuffer', 'LongInt'( 1 );
6946: 'TLSerror_InvalidParameter', 'LongInt'( 2 );
6947: 'TLSerror_InvalidCertificate', 'LongInt'( 3 );
6948: 'TLSerror_InvalidState', 'LongInt'( 4 );
6949: 'TLSerror_DecodeError', 'LongInt'( 5 );
6950: 'TLSerror_BadProtocol', 'LongInt'( 6 );
6951: Function TLSErrorMessage( const TLSError : Integer ) : String
6952: SIRRegister_ETLSError(CL);
6953: TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6954: PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6955: Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6956: Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6957: Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6958: Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6959: Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion ) : Boolean
6960: Function IsSSL2( const A : TTLSProtocolVersion ) : Boolean
6961: Function IsSSL3( const A : TTLSProtocolVersion ) : Boolean
6962: Function IsTLS10( const A : TTLSProtocolVersion ) : Boolean
6963: Function IsTLS11( const A : TTLSProtocolVersion ) : Boolean
6964: Function IsTLS12( const A : TTLSProtocolVersion ) : Boolean
6965: Function IsTLS10OrLater( const A : TTLSProtocolVersion ) : Boolean
6966: Function IsTLS11OrLater( const A : TTLSProtocolVersion ) : Boolean
6967: Function IsTLS12OrLater( const A : TTLSProtocolVersion ) : Boolean
6968: Function IsFutureTLSVersion( const A : TTLSProtocolVersion ) : Boolean
6969: Function IsKnownTLSVersion( const A : TTLSProtocolVersion ) : Boolean
6970: Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion ) : String
6971: Function TLSProtocolVersionName( const A : TTLSProtocolVersion ) : String
6972: PTLSRandom', '^PTLSRandom // will not work
6973: Procedure InitTLSRandom( var Random : PTLSRandom )
6974: Function TLSRandomToStr( const Random : PTLSRandom ) : AnsiString
6975: 'TLSSessionIDMaxLen', 'LongInt'( 32 );
6976: Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString )
6977: Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6978: Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6979: TTLSignatureAndHashAlgorithm', 'record Hash : TTLSShashAlgorithm'
6980: +' ; Signature : TTLSSignatureAlgorithm; end
6981: // PTLSsignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6982: TTLSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6983: TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_
6984: +' DSS, tlskeadHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )

```

```

6985: TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6986: +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )'
6987: TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : Integer;
6988: +'integer; Supported : Boolean; end'
6989: PTLSMacAlgorithmInfo', '^TTLSSMacAlgorithmInfo // will not work
6990: 'TLS_MAC_MAXDIGESTSIZE', 'LongInt'( 64 );
6991: TTLSPRFAlgorithm', '( tlspaSHA256 )
6992: Function tlsp_MD5( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6993: Function tlsp_SHA1( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6994: Function tlsp_SHA256( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6995: Function tlsp_SHA512( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6996: Function tlsp1OPRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer ) : AnsiString
6997: Function tlsp12PRF_SHA256( const Secret, ALLabel, Seed : AnsiString; const Size : Integer ) : AnsiString
6998: Function tlsp12PRF_SHA512( const Secret, ALLabel, Seed : AnsiString; const Size : Integer ) : AnsiString
6999: Function TLSPRF( const ProtoVersion:TTLSSProtocolVersion;const Secret,ALabel,Seed:AString;const
Size:Int):AString;
7000: Function tlsp10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
Size:Integer):AnsiString
7001: Function tlsp12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
7002: Function tlsp12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
7003: Function TLSKeyBlock(const ProtocolVersion:TTLSSProtocolVersion; const MasterSecret, ServerRandom,
ClientRandom : AnsiString; const Size : Integer ) : AnsiString
7004: Function tlsp10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
7005: Function tlsp12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
7006: Function tlsp12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
7007: Function TLSMasterSecret( const ProtocolVersion: TTLSSProtocolVersion;const PreMasterSecret,ClientRandom,
ServerRandom:AnsiString ) : AnsiString
7008: 'TLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr
7009: +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
7010: +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
7011: Procedure GenerateTLSKeys( const ProtocolVersion : TTLSSProtocolVersion; const MACKeyBits, CipherKeyBits,
IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSSKeys )
7012: Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSSProtocolVersion;const IsExportable:Boolean;const
ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSSKeys)
7013: 'TLS_PLAINTEXT_FRAGMENT_MAXSIZE', 'LongInt'( 16384 - 1 );
7014: 'TLS_COMPRESSED_FRAGMENT_MAXSIZE', 'LongInt'( 16384 + 1024 );
7015: Procedure SelfTestcTLSUtils
7016: end;
7017:
7018: (*-----*)
7019: procedure SIRegister_Reversi(CL: TPSPPascalCompiler);
7020: begin
7021:   sPosData', 'record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
integer; disks : integer; mx : integer; my : integer; end
7022: // pBoard', '^pBoard // will not work
7023: Function rCalculateData( cc : byte; cx, cy : integer ) : sPosData
7024: Function rCheckMove( color : byte; cx, cy : integer ) : integer
7025: //Function rDoStep( data : pBoard ) : word
7026: Function winExecAndWait( const sAppPath : string; wVisible : word ) : boolean
7027: end;
7028:
7029: procedure SIRegister_IWDBCommon(CL: TPSPPascalCompiler);
7030: begin
7031:   Function InEditMode( ADataSet : TDataSet ) : Boolean
7032:   Function CheckDataSource( ADataSource : TDataSource ) : Boolean;
7033:   Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
7034:   Function GetFieldText( AField : TField ) : String
7035: end;
7036:
7037: procedure SIRegister_SortGrid(CL: TPSPPascalCompiler);
7038: begin
7039:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7040:   TMyPrintRange', '( prAll, prSelected )
7041:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7042:   +'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'
7046:   +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7047:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7048:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7049:   SIRegister_TSortOptions(CL);
7050:   SIRegister_TPrintOptions(CL);
7051:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7052:   SIRegister_TSortedList(CL);
7053:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7054:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7055:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7056:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7057:   +'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 : '
7061:   +' integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7062:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7063:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7064:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)

```

```

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

```

```

7153: Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex ) : Cardinal
7154: Function XRTLSwapHiLo16( X : Word ) : Word
7155: Function XRTLSwapHiLo32( X : Cardinal ) : Cardinal
7156: Function XRTLSwapHiLo64( X : Int64 ) : Int64
7157: Function XRTLROL32( A, S : Cardinal ) : Cardinal
7158: Function XRTLROL32( A, S : Cardinal ) : Cardinal
7159: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7160: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7161: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7162: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7163: //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer )
7164: //Procedure XRTLIncBlock( P : PByteArray; Len : integer )
7165: Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer )
7166: Function XRTLPopulation( A : Cardinal ) : Cardinal
7167: end;
7168:
7169: Function XRTLURLDecode( const ASrc : WideString ) : WideString
7170: Function XRTLURLEncode( const ASrc : WideString ) : WideString
7171: Function XRTLURINormalize( const AURI : WideString ) : WideString
7172: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,VPassword : WideString)
7173: Function XRTLExtractLongPathName(APath: string): string;
7174:
7175: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7176: begin
7177:   Int8', 'ShortInt
7178:   AddTypeS('Int16', 'SmallInt
7179:   Int32', 'LongInt
7180:   UInt8', 'Byte
7181:   UInt16', 'Word
7182:   UInt32', 'LongWord
7183:   UInt64', 'Int64
7184:   Word8', 'UInt8
7185:   Word16', 'UInt16
7186:   Word32', 'UInt32
7187:   Word64', 'UInt64
7188:   LargeInt', 'Int64
7189:   NativeInt', 'Integer
7190:   AddTypes('NativeUInt', 'Cardinal
7191:   Const('BitsPerByte','LongInt'( 8);
7192:   Const('BitsPerWord','LongInt'( 16);
7193:   Const('BitsPerLongWord','LongInt'( 32);
7194:   //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7195:   //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7196:   Function MinI( const A, B : Integer ) : Integer
7197:   Function MaxI( const A, B : Integer ) : Integer
7198:   Function MinC( const A, B : Cardinal ) : Cardinal
7199:   Function MaxC( const A, B : Cardinal ) : Cardinal
7200:   Function SumClipI( const A, I : Integer ) : Integer
7201:   Function SumClipC( const A : Cardinal; const I : Integer ) : Cardinal
7202:   Function InByteRange( const A : Int64 ) : Boolean
7203:   Function InWordRange( const A : Int64 ) : Boolean
7204:   Function InLongWordRange( const A : Int64 ) : Boolean
7205:   Function InShortIntRange( const A : Int64 ) : Boolean
7206:   Function InSmallIntRange( const A : Int64 ) : Boolean
7207:   Function InLongIntRange( const A : Int64 ) : Boolean
7208:   AddTypeS('Bool8', 'ByteBool
7209:   AddTypeS('Bool16', 'WordBool
7210:   AddTypeS('Bool32', 'LongBool
7211:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7212:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7213:   Function ReverseCompareResult( const C : TCompareResult ) : TCompareResult
7214:   Const('MinSingle','Single').setExtended( 1.5E-45 );
7215:   Const('MaxSingle','Single').setExtended( 3.4E+38 );
7216:   Const('MinDouble','Double').setExtended( 5.0E-324 );
7217:   Const('MaxDouble','Double').setExtended( 1.7E+308 );
7218:   Const('MinExtended','Extended').setExtended(3.4E-4932);
7219:   Const('MaxExtended','Extended').setExtended(1.1E+4932);
7220:   Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807 );
7221:   Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807 );
7222:   Function MinF( const A, B : Float ) : Float
7223:   Function MaxF( const A, B : Float ) : Float
7224:   Function ClipF( const Value : Float; const Low, High : Float ) : Float
7225:   Function InSingleRange( const A : Float ) : Boolean
7226:   Function InDoubleRange( const A : Float ) : Boolean
7227:   Function InCurrencyRange( const A : Float ) : Boolean;
7228:   Function InCurrencyRange1( const A : Int64 ) : Boolean;
7229:   Function FloatExponentBase2( const A : Extended; var Exponent : Integer ) : Boolean
7230:   Function FloatExponentBase10( const A : Extended; var Exponent : Integer ) : Boolean
7231:   Function FloatIsInfinity( const A : Extended ) : Boolean
7232:   Function FloatIsNaN( const A : Extended ) : Boolean
7233:   Const('SingleCompareDelta','Extended').setExtended( 1.0E-34 );
7234:   Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280 );
7235:   Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400 );
7236:   Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34 );
7237:   Function FloatZero( const A : Float; const CompareDelta : Float ) : Boolean
7238:   Function FloatOne( const A : Float; const CompareDelta : Float ) : Boolean
7239:   Function FloatsEqual( const A, B : Float; const CompareDelta : Float ) : Boolean
7240:   Function FloatsCompare( const A, B : Float; const CompareDelta : Float ) : TCompareResult

```

```

7241: Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7242: Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7243: Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7244: Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7245: Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7246: Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7247: Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7248: Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7249: Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7250: Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7251: Function cIsHighBitSet( const Value : LongWord) : Boolean
7252: Function SetBitScanForward( const Value : LongWord) : Integer;
7253: Function SetBitScanForward1( const Value : LongWord) : Integer;
7254: Function SetBitScanReverse( const Value : LongWord) : Integer;
7255: Function SetBitScanReverse1( const Value : LongWord) : Integer;
7256: Function ClearBitScanForward( const Value : LongWord) : Integer;
7257: Function ClearBitScanForward1( const Value : LongWord) : Integer;
7258: Function ClearBitScanReverse( const Value : LongWord) : Integer;
7259: Function ClearBitScanReverse1( const Value : LongWord) : Integer;
7260: Function cReverseBits( const Value : LongWord) : LongWord;
7261: Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7262: Function cSwapEndian( const Value : LongWord) : LongWord
7263: Function cTwosComplement( const Value : LongWord) : LongWord
7264: Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7265: Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7266: Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7267: Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7268: Function cBitCount( const Value : LongWord) : LongWord
7269: Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7270: Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7271: Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7272: Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7273: Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7274: Function ClearBitRange(const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7275: Function ToggleBitRange(const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7276: Function IsBitRangeSet(const Value: LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7277: Function IsBitRangeClear(const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : Boolean
7278: // AddTypeS(' CharSet', 'set of AnsiChar'
7279: AddTypeS(' CharSet', 'set of Char' //!!!
7280: AddTypeS(' AnsiCharSet', 'TCharSet
7281: AddTypeS(' ByteSet', 'set of Byte
7282: AddTypeS(' AnsiChar', 'Char
7283: // Function AsCharSet( const C : array of AnsiChar) : CharSet
7284: Function AsByteSet( const C : array of Byte) : ByteSet
7285: Procedure ComplementChar( var C : CharSet; const Ch : Char)
7286: Procedure ClearCharSet( var C : CharSet)
7287: Procedure FillCharSet( var C : CharSet)
7288: procedure FillCharSearchRec; // with 0
7289: Procedure ComplementCharSet( var C : CharSet)
7290: Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7291: Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7292: Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7293: Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7294: Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7295: Function IsSubSet( const A, B : CharSet) : Boolean
7296: Function IsEqual( const A, B : CharSet) : Boolean
7297: Function IsEmpty( const C : CharSet) : Boolean
7298: Function IsComplete( const C : CharSet) : Boolean
7299: Function cCharCount( const C : CharSet) : Integer
7300: Procedure ConvertCaseInsensitive( var C : CharSet)
7301: Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7302: Function IntRangeLength( const Low, High : Integer) : Int64
7303: Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7304: Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7305: Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7306: Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7307: Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7308: Function CardRangeLength( const Low, High : Cardinal) : Int64
7309: Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7310: Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7311: Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7312: Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7313: Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7314: AddTypeS(' UnicodeChar', 'WideChar
7315: Function Compare( const I1, I2 : Boolean) : TCompareResult;
7316: Function CompareI( const I1, I2 : Integer) : TCompareResult;
7317: Function Compare2( const I1, I2 : Int64) : TCompareResult;
7318: Function Compare3( const I1, I2 : Extended) : TCompareResult;
7319: Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7320: Function CompareW( const I1, I2 : WideString) : TCompareResult
7321: Function cSgn( const A : LongInt) : Integer;
7322: Function cSgn1( const A : Int64) : Integer;
7323: Function cSgn2( const A : Extended) : Integer;
7324: AddTypeS(' TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7325: Function AnsiCharToInt( const A : AnsiChar) : Integer
7326: Function WideCharToInt( const A : WideChar) : Integer
7327: Function CharToInt( const A : Char) : Integer
7328: Function IntToAnsiChar( const A : Integer) : AnsiChar
7329: Function IntToWideChar( const A : Integer) : WideChar

```

```

7330: Function IntToChar( const A : Integer ) : Char
7331: Function IsHexAnsiChar( const Ch : AnsiChar ) : Boolean
7332: Function IsHexWideChar( const Ch : WideChar ) : Boolean
7333: Function IsHexChar( const Ch : Char ) : Boolean
7334: Function HexAnsiCharToInt( const A : AnsiChar ) : Integer
7335: Function HexWideCharToInt( const A : WideChar ) : Integer
7336: Function HexCharToInt( const A : Char ) : Integer
7337: Function IntToUpperHexAnsiChar( const A : Integer ) : AnsiChar
7338: Function IntToUpperHexWideChar( const A : Integer ) : WideChar
7339: Function IntToUpperHexChar( const A : Integer ) : Char
7340: Function IntToLowerHexAnsiChar( const A : Integer ) : AnsiChar
7341: Function IntToLowerHexWideChar( const A : Integer ) : WideChar
7342: Function IntToLowerHexChar( const A : Integer ) : Char
7343: Function IntToStringA( const A : Int64 ) : AnsiString
7344: Function IntToStringW( const A : Int64 ) : WideString
7345: Function IntToString( const A : Int64 ) : String
7346: Function UIntToStringA( const A : NativeUInt ) : AnsiString
7347: Function UIntToStringW( const A : NativeUInt ) : WideString
7348: Function UIntToString( const A : NativeUInt ) : String
7349: Function LongWordToStrA( const A : LongWord; const Digits : Integer ) : AnsiString
7350: Function LongWordToStrW( const A : LongWord; const Digits : Integer ) : WideString
7351: Function LongWordToStrU( const A : LongWord; const Digits : Integer ) : UnicodeString
7352: Function LongWordToStr( const A : LongWord; const Digits : Integer ) : String
7353: Function LongWordToHexA( const A : LongWord; const Digits : Integer; const UpperCase : Boolean ) : AnsiString
7354: Function LongWordToHexW( const A : LongWord; const Digits : Integer; const UpperCase : Boolean ) : WideString
7355: Function LongWordToHex( const A : LongWord; const Digits : Integer; const UpperCase : Boolean ) : String
7356: Function LongWordToOctA( const A : LongWord; const Digits : Integer ) : AnsiString
7357: Function LongWordToOctW( const A : LongWord; const Digits : Integer ) : WideString
7358: Function LongWordToOct( const A : LongWord; const Digits : Integer ) : String
7359: Function LongWordToBinA( const A : LongWord; const Digits : Integer ) : AnsiString
7360: Function LongWordToBinW( const A : LongWord; const Digits : Integer ) : WideString
7361: Function LongWordToBin( const A : LongWord; const Digits : Integer ) : String
7362: Function TryStringToInt64A( const S : AnsiString; out A : Int64 ) : Boolean
7363: Function TryStringToInt64W( const S : WideString; out A : Int64 ) : Boolean
7364: Function TryStringToInt64( const S : String; out A : Int64 ) : Boolean
7365: Function StringToInt64DefA( const S : AnsiString; const Default : Int64 ) : Int64
7366: Function StringToInt64DefW( const S : WideString; const Default : Int64 ) : Int64
7367: Function StringToInt64Def( const S : String; const Default : Int64 ) : Int64
7368: Function StringToInt64A( const S : AnsiString ) : Int64
7369: Function StringToInt64W( const S : WideString ) : Int64
7370: Function StringToInt64( const S : String ) : Int64
7371: Function TryStringToIntA( const S : AnsiString; out A : Integer ) : Boolean
7372: Function TryStringToIntW( const S : WideString; out A : Integer ) : Boolean
7373: Function TryStringToInt( const S : String; out A : Integer ) : Boolean
7374: Function StringToIntDefA( const S : AnsiString; const Default : Integer ) : Integer
7375: Function StringToIntDefW( const S : WideString; const Default : Integer ) : Integer
7376: Function StringToIntDef( const S : String; const Default : Integer ) : Integer
7377: Function StringToIntA( const S : AnsiString ) : Integer
7378: Function StringToIntW( const S : WideString ) : Integer
7379: Function StringToInt( const S : String ) : Integer
7380: Function TryStringToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7381: Function TryStringToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7382: Function TryStringToLongWord( const S : String; out A : LongWord ) : Boolean
7383: Function StringToLongWordA( const S : AnsiString ) : LongWord
7384: Function StringToLongWordW( const S : WideString ) : LongWord
7385: Function StringToLongWord( const S : String ) : LongWord
7386: Function HexToUIntA( const S : AnsiString ) : NativeUInt
7387: Function HexToUIntW( const S : WideString ) : NativeUInt
7388: Function HexToUInt( const S : String ) : NativeUInt
7389: Function TryHexToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7390: Function TryHexToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7391: Function TryHexToLongWord( const S : String; out A : LongWord ) : Boolean
7392: Function HexToLongWordA( const S : AnsiString ) : LongWord
7393: Function HexToLongWordW( const S : WideString ) : LongWord
7394: Function HexToLongWord( const S : String ) : LongWord
7395: Function TryOctToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7396: Function TryOctToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7397: Function TryOctToLongWord( const S : String; out A : LongWord ) : Boolean
7398: Function OctToLongWordA( const S : AnsiString ) : LongWord
7399: Function OctToLongWordW( const S : WideString ) : LongWord
7400: Function OctToLongWord( const S : String ) : LongWord
7401: Function TryBinToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7402: Function TryBinToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7403: Function TryBinToLongWord( const S : String; out A : LongWord ) : Boolean
7404: Function BinToLongWordA( const S : AnsiString ) : LongWord
7405: Function BinToLongWordW( const S : WideString ) : LongWord
7406: Function BinToLongWord( const S : String ) : LongWord
7407: Function FloatToStringA( const A : Extended ) : AnsiString
7408: Function FloatToStringW( const A : Extended ) : WideString
7409: Function FloatToString( const A : Extended ) : String
7410: Function TryStringToFloatA( const A : AnsiString; out B : Extended ) : Boolean
7411: Function TryStringToFloatW( const A : WideString; out B : Extended ) : Boolean
7412: Function TryStringToFloat( const A : String; out B : Extended ) : Boolean
7413: Function StringToFloatA( const A : AnsiString ) : Extended
7414: Function StringToFloatW( const A : WideString ) : Extended
7415: Function StringToFloat( const A : String ) : Extended
7416: Function StringToFloatDefA( const A : AnsiString; const Default : Extended ) : Extended
7417: Function StringToFloatDefW( const A : WideString; const Default : Extended ) : Extended
7418: Function StringToFloatDef( const A : String; const Default : Extended ) : Extended

```

```

7419: Function EncodeBase64( const S:Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
    PadChar: AnsiChar ) : AnsiString
7420: Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet ) : AnsiString
7421: unit uPSI_cfundamentUtils;
7422: Const('b64_MIMEBase64','Str').String('ABCDEFIGHIJKLNMOPQRSTUVWXYZZabedefghijklnopqrstuvwxyz0123456789+/-');
7423: Const('b64_UUEncode','String').String('!"#$%&"()*+,.-./0123456789:;<>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[\]^_');
7424: Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNPQRSTUVWXYZZabedefghijklnopqrstuvwxyz');
7425: Const('CCHARSET','Stringb64_XXEncode');
7426: Const('CHEXSET','String'0123456789ABCDEF
7427: Const('HEXDIGITS','String'0123456789ABCDEF
7428: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7429: Const('DIGISET','String'0123456789
7430: Const('LETTERSET','String'ABCDEFIGHIJKLMNOPQRSTUVWXYZ'
7431: Const('DIGISET2','TCharset').SetSet('0123456789'
7432: Const('LETTERSET2','TCharset').SetSet('ABCDEFIGHIJKLMNOPQRSTUVWXYZ')
7433: Const('HEXSET2','TCharset').SetSet('0123456789ABCDEF');
7434: Const('NUMBERSET','TCharset').SetSet('0123456789');
7435: Const('NUMBERS','String'0123456789);
7436: Const('LETTERS','String'ABCDEFIGHIJKLMNOPQRSTUVWXYZ');
7437: Const('NUMBLETTSET','String').SetString('0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ');
7438: Const('NUMBLETTSET','TCharset').SetSet('0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ');
7439: Function CharSetToStr( const C : CharSet ) : AnsiString
7440: Function StrToCharSet( const S : AnsiString ) : CharSet
7441: Function MIMEBase64Decode( const S : AnsiString ) : AnsiString
7442: Function MIMEBase64Encode( const S : AnsiString ) : AnsiString
7443: Function UUDecode( const S : AnsiString ) : AnsiString
7444: Function XXDecode( const S : AnsiString ) : AnsiString
7445: Function BytesToHex( const P : array of Byte; const UpperCase : Boolean ) : AnsiString
7446: Function InterfaceToStrA( const I : IInterface ) : WideString
7447: Function InterfaceToStrW( const I : IInterface ) : WideString
7448: Function InterfaceToStr( const I : IInterface ) : String
7449: Function ObjectClassName( const O : TObject ) : String
7450: Function ClassClassName( const C : TClass ) : String
7451: Function ObjectToStr( const O : TObject ) : String
7452: Function ObjectToString( const O : TObject ) : String
7453: Function CharSetToStr( const C : CharSet ) : AnsiString
7454: Function StrToCharSet( const S : AnsiString ) : CharSet
7455: Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
    AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7456: Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
    AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7457: Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
    : Boolean; const Slots : LongWord) : LongWord
7458: Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7459: Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7460: Const('Bytes1KB','LongInt'( 1024));
7461: SIRegister_IInterface(CL);
7462: Procedure SelfTestCFundamentUtils
7463:
7464: Function CreateSchedule : IJclSchedule
7465: Function NullStamp : TTimeStamp
7466: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Int64
7467: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Boolean
7468: Function IsNullTimeStamp( const Stamp : TTimeStamp ) : Boolean
7469:
7470: procedure SIRegister_uwinplot(CL: TPPSPascalCompiler);
7471: begin
7472:   AddTypeS('TFunc', 'function(X : Float) : Float;');
7473:   Function InitGraphics( Width, Height : Integer ) : Boolean
7474:   Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean )
7475:   Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float )
7476:   Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float )
7477:   Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float )
7478:   Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float )
7479:   Procedure SetGraphTitle( Title : String )
7480:   Procedure SetOxTitle( Title : String )
7481:   Procedure SetOyTitle( Title : String )
7482:   Function GetGraphTitle : String
7483:   Function GetOxTitle : String
7484:   Function GetOyTitle : String
7485:   Procedure PlotOxAxis( Canvas : TCanvas )
7486:   Procedure PlotOyAxis( Canvas : TCanvas )
7487:   Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid )
7488:   Procedure WriteGraphTitle( Canvas : TCanvas )
7489:   Function SetMaxCurv( NCurv : Byte ) : Boolean
7490:   Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor )
7491:   Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor )
7492:   Procedure SetCurvLegend( CurvIndex : Integer; Legend : String )
7493:   Procedure SetCurvStep( CurvIndex, Step : Integer )
7494:   Function GetMaxCurv : Byte
7495:   Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor )
7496:   Procedure GetLineParam(CurvIndex:Intger;var Style:TPenStyle;var Width:Intger; var Color:TColor);
7497:   Function GetCurvLegend( CurvIndex : Integer ) : String
7498:   Function GetCurvStep( CurvIndex : Integer ) : Integer
7499:   Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer )
7500:   Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer )
7501:   Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7502:   Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7503:   Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean )

```

```

7504: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7505: Function Xpixel( X : Float ) : Integer
7506: Function Ypixel( Y : Float ) : Integer
7507: Function Xuser( X : Integer ) : Float
7508: Function Yuser( Y : Integer ) : Float
7509: end;
7510:
7511: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7512: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7513: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7514: Procedure FFT_Integer_Cleanup
7515: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7516: //unit uPSI_JclStreams;
7517: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin ) : Int64
7518: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer ) : Int64
7519: Function CompareStreams( A, B : TStream; BufferSize : Integer ) : Boolean
7520: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer ) : Boolean
7521:
7522: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7523: begin
7524:   FindClass('TOBJECT','EInvalidDest'
7525:   FindClass('TOBJECT','EFCantMove'
7526:   Procedure fmxCopyFile( const FileName, DestName : string )
7527:   Procedure fmxMoveFile( const FileName, DestName : string )
7528:   Function fmxGetFileSize( const FileName : string ) : LongInt
7529:   Function fmxFileDateTime( const FileName : string ) : TDateTime
7530:   Function fmxHasAttr( const FileName : string; Attr : Word ) : Boolean
7531:   Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ):THandle;
7532: end;
7533:
7534: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7535: begin
7536:   SIRegister_IFindFileIterator(CL);
7537:   Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7538: end;
7539:
7540: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7541: begin
7542:   Function SkipWhite( cp : PChar ) : PChar
7543:   Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string ) : PChar
7544:   Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string ) : PChar
7545:   Function ReadIdent( cp : PChar; var ident : string ) : PChar
7546: end;
7547:
7548: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7549: begin
7550:   SIRegister_TStringHashMapTraits(CL);
7551:   Function CaseSensitiveTraits : TStringHashMapTraits
7552:   Function CaseInsensitiveTraits : TStringHashMapTraits
7553:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7554:   +'e; Right : PHashNode; end
7555:   //PHashArray', '^THashArray // will not work
7556:   SIRegister_TStringHashMap(CL);
7557:   THashValue', 'Cardinal
7558:   Function StrHash( const s : string ) : THashValue
7559:   Function TextHash( const s : string ) : THashValue
7560:   Function DataHash( var AValue, ASize : Cardinal ) : THashValue
7561:   Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer ) : Boolean
7562:   Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer ) : Boolean
7563:   Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer ) : Boolean
7564:   SIRegister_TCaseSensitiveTraits(CL);
7565:   SIRegister_TCaseInsensitiveTraits(CL);
7566:
7567:
7568: //*****unit uPSI_umath;
7569: Function uExpo( X : Float ) : Float
7570: Function uExp2( X : Float ) : Float
7571: Function uExp10( X : Float ) : Float
7572: Function uLog( X : Float ) : Float
7573: Function uLog2( X : Float ) : Float
7574: Function uLog10( X : Float ) : Float
7575: Function uLogA( X, A : Float ) : Float
7576: Function uIntPower( X : Float; N : Integer): Float
7577: Function uPower( X, Y : Float ) : Float
7578: Function SgnGamma( X : Float ) : Integer
7579: Function Stirling( X : Float ) : Float
7580: Function StirLog( X : Float ) : Float
7581: Function Gamma( X : Float ) : Float
7582: Function LnGamma( X : Float ) : Float
7583: Function DiGamma( X : Float ) : Float
7584: Function TriGamma( X : Float ) : Float
7585: Function IGamma( X : Float ) : Float
7586: Function JGamma( X : Float ) : Float
7587: Function InvGamma( X : Float ) : Float
7588: Function Erf( X : Float ) : Float
7589: Function Erfc( X : Float ) : Float
7590: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7591: { Correlation coefficient between samples X and Y }
7592: function DBeta(A, B, X : Float) : Float;

```

```

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

```

```

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

```

```

7768: Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7769: Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7770: Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7771: Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7772: Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7773: Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7774: Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7775: Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7776: Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7777: Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7778: Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7779: Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7780: Procedure Emboss( var Bmp : TBitmap)
7781: Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7782: Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7783: Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7784: Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7785: Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7786: Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7787: Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7788: Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7789: Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7790: Procedure QuartoOpaque( Src, Dst : TBitmap)
7791: Procedure SemiOpaque( Src, Dst : TBitmap)
7792: Procedure ShadowDownLeft( const Dst : TBitmap)
7793: Procedure ShadowDownRight( const Dst : TBitmap)
7794: Procedure ShadowUpLeft( const Dst : TBitmap)
7795: Procedure ShadowUpRight( const Dst : TBitmap)
7796: Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7797: Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7798: Procedure FlipRight( const Dst : TBitmap)
7799: Procedure FlipDown( const Dst : TBitmap)
7800: Procedure Spotlight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7801: Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7802: Procedure MakeSeamlessClip( var Dst : TBitmap; Sean : Integer)
7803: Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7804: Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7805: Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7806: Procedure SmoothResize( var Src, Dst : TBitmap)
7807: Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7808: Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7809: Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7810: Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7811: Procedure GrayScale( const Dst : TBitmap)
7812: Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7813: Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7814: Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7815: Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7816: Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7817: Procedure Spray( const Dst : TBitmap; Amount : Integer)
7818: Procedure AntiAlias( const Dst : TBitmap)
7819: Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7820: Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7821: Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7822: Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7823: Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7824: Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7825: Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7826: Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7827: Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7828: Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7829: Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7830: Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7831: Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7832: Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7833: Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7834: Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7835: Procedure Invert( Src : TBitmap)
7836: Procedure MirrorRight( Src : TBitmap)
7837: Procedure MirrorDown( Src : TBitmap)
7838: end;
7839: end;
7840:
7841: (*-----*)
7842: procedure SJRegister_JvPaintFX(CL: TPSPPascalCompiler);
7843: begin
7844:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7845:             +'ye, lbtwist, lbrimble, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7846:             +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )'
7847:   SJRegister_TJvPaintFX(CL);
7848:   Function SplineFilter( Value : Single) : Single
7849:   Function BellFilter( Value : Single) : Single
7850:   Function TriangleFilter( Value : Single) : Single
7851:   Function BoxFilter( Value : Single) : Single
7852:   Function HermiteFilter( Value : Single) : Single
7853:   Function Lanczos3Filter( Value : Single) : Single
7854:   Function MitchellFilter( Value : Single) : Single
7855: end;
7856:

```

```

7857:
7858: (*-----*)
7859: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7860: begin
7861:   'TeeMsg_DefaultFunctionName', 'String 'TeeFunction
7862:   TeeMsg_DefaultSeriesName', 'String 'Series
7863:   TeeMsg_DefaultToolName', 'String 'ChartTool
7864:   ChartComponentPalette', 'String 'TeeChart
7865:   TeeMaxLegendColumns', 'LongInt'( 2 );
7866:   TeeDefaultLegendSymbolWidth', 'LongInt'( 20 );
7867:   TeeTitleFootDistance, LongInt( 5 );
7868:   SIRegister_TCustomChartWall(CL);
7869:   SIRegister_TChartWall(CL);
7870:   SIRegister_TChartLegendGradient(CL);
7871:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7872:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7873:   FindClass('TOBJECT'), 'LegendException
7874:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7875:     +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7876:   FindClass('TOBJECT'), 'TCustomChartLegend
7877:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7878:   TLegendSymbolPosition', '( spLeft, spRight )
7879:   TSymbolDrawEvent', 'Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect)';
7880:   TSymbolCalcHeight', 'Function : Integer
7881:   SIRegister_TLegendSymbol(CL);
7882:   SIRegister_TTeeCustomShapePosition(CL);
7883:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7884:   SIRegister_TLegendTitle(CL);
7885:   SIRegister_TLegendItem(CL);
7886:   SIRegister_TLegendItems(CL);
7887:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7888:   FindClass('TOBJECT'), 'TCustomChart
7889:   SIRegister_TCustomChartLegend(CL);
7890:   SIRegister_TChartLegend(CL);
7891:   SIRegister_TChartTitle(CL);
7892:   SIRegister_TChartFootTitle(CL);
7893:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7894:     +'eButton; Shift : TShiftState; X, Y : Integer)
7895:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7896:     +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7897:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series : '
7898:     +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7899:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7900:     +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7901:   TOnGetLegendPos', 'Procedure ( Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7902:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7903:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7904:     +'toMax : Boolean; Min : Double; Max : Double; end
7905:   TA11AxisSavedScales', 'array of TAxisSavedScales
7906:   SIRegister_TChartBackWall(CL);
7907:   SIRegister_TChartRightWall(CL);
7908:   SIRegister_TChartBottomWall(CL);
7909:   SIRegister_TChartLeftWall(CL);
7910:   SIRegister_TChartWalls(CL);
7911:   TChartAllowScrollEvent', 'Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean)';
7912:   SIRegister_TCustomChart(CL);
7913:   SIRegister_TChart(CL);
7914:   SIRegister_TTeeSeriesTypes(CL);
7915:   SIRegister_TTeeToolTypes(CL);
7916:   SIRegister_TTeeDragObject(CL);
7917:   SIRegister_TColorPalettes(CL);
7918:   Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
    AGalleryPage:PString;ANumGallerySeries:Integer;
7919:   Procedure RegisterTeeSeriesl( ASeriesClass : TChartSeriesClass; ADscription : PString);
7920:   Procedure RegisterTeeFunction(AFuncfClass:TTeeFunctionClass;ADescription,
    AGalleryPage:PString;ANumGallerySeries : Int;
7921:   Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADscription : PString)
7922:   Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
    ADscription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7923:   Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7924:   Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7925:   Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7926:   Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7927:   Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
    AFunctionClass : TTeeFunctionClass) : TChartSeries
7928:   Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7929:   Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7930:   Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7931:   Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent ) : TTeeCustomTool
7932:   Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7933:   Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7934:   Function GetNewSeriesName( AOwner : TComponent ) : TComponentName
7935:   Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7936:   Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7937:   Function GetGallerySeriesName( ASeries : TChartSeries) : String
7938:   Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
    R:TRect;ReferenceChart:TCustomChart);
7939:   SIRegister_TChartTheme(CL);
7940:   //TChartThemeClass', 'class of TChartTheme

```

```

7941: //TCanvasClass', 'class of TCanvas3D
7942: Function SeriesNameOrIndex( ASeries : TCustomChartSeries ) : String
7943: Function SeriesTitleOrName( ASeries : TCustomChartSeries ) : String
7944: Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean )
7945: Procedure ShowMessageUser( const S : String )
7946: Function HasNoMandatoryValues( ASeries : TChartSeries ) : Boolean
7947: Function HasLabels( ASeries : TChartSeries ) : Boolean
7948: Function HasColors( ASeries : TChartSeries ) : Boolean
7949: Function SeriesGuessContents( ASeries : TChartSeries ) : TeeFormatFlag
7950: Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer )
7951: end;
7952:
7953:
7954: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7955: begin
7956: // 'TeeFormBorderStyle',' bsNone );
7957: SIRegister_TMetafile(CL);
7958: 'TeeDefVerticalMargin','LongInt'( 4 );
7959: 'TeeDefHorizMargin','LongInt'( 3 );
7960: 'crTeeHand','LongInt'( TCursor( 2020 ) );
7961: 'TeeMsg_TeeHand','String 'crTeeHand
7962: 'TeeNormalPrintDetail','LongInt'( 0 );
7963: 'TeeHighPrintDetail','LongInt'( -100 );
7964: 'TeeDefault_PrintMargin','LongInt'( 15 );
7965: 'MaxDefaultColors','LongInt'( 19 );
7966: 'TeeTabDelimiter','Char #9';
7967: TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7968: +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7969: +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7970: +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7971: +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7972: +'ourMonths, dtSixMonths, dtOneYear, dtNone )'
7973: SIRegister_TCustomPanelNoCaption(CL);
7974: FindClass('TOBJECT'), 'TCustomTeePanel
7975: SIRegister_TZoomPanning(CL);
7976: SIRegister_TTeeEvent(CL);
7977: //SIRegister_TTeeEventListeners(CL);
7978: TTeeMouseEventKind', '( meDown, meUp, meMove )'
7979: SIRegister_TTeeMouseEvent(CL);
7980: SIRegister_TCustomTeePanel(CL);
7981: //TChartGradient', 'TTeeGradient
7982: //TChartGradientClass', 'class of TChartGradient
7983: TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )'
7984: SIRegister_TTeeZoomPen(CL);
7985: SIRegister_TTeeZoomBrush(CL);
7986: TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )'
7987: SIRegister_TTeeZoom(CL);
7988: FindClass('TOBJECT'), 'TCustomTeePanelExtended
7989: TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )'
7990: SIRegister_TBackImage(CL);
7991: SIRegister_TCustomTeePanelExtended(CL);
7992: //TChartBrushClass', 'class of TChartBrush
7993: SIRegister_TTeeCustomShapeBrushPen(CL);
7994: TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )'
7995: TTextFormat', '( ttfNormal, ttfHtml )'
7996: SIRegister_TTeeCustomShape(CL);
7997: SIRegister_TTeeShape(CL);
7998: SIRegister_TTeeExportData(CL);
7999: Function TeeStr( const Num : Integer ) : String
8000: Function DateTimeDefaultFormat( const AStep : Double ) : String
8001: Function TEEDaysInMonth( Year, Month : Word ) : Word
8002: Function FindDateTimeStep( const StepValue : Double ) : TDateTimeStep
8003: Function NextDateTimeStep( const AStep : Double ) : Double
8004: Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer ) : Boolean;
8005: Function PointInLine1( const P, FromPoint, ToPoint : TPoint ) : Boolean;
8006: Function PointInLine2( const P, FromPoint, ToPoint : TPoint; const TolerancePixels : Integer ) : Boolean;
8007: Function PointInLine3( const P : TPoint; const px, py, qx, qy : Integer ) : Boolean;
8008: Function PointInLineTolerance( const P : TPoint; const px, py, qx, qy, TolerancePixels : Integer ) : Boolean;
8009: Function PointInPolygon( const P : TPoint; const Poly : array of TPoint ) : Boolean;
8010: Function PointInTriangle( const P, P0, P1, P2 : TPoint ) : Boolean;
8011: Function PointInTriangle1( const P : TPoint; X0, Y0, X1, Y1 : Integer ) : Boolean;
8012: Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer ) : Boolean;
8013: Function PointInEllipse( const P : TPoint; const Rect : TRect ) : Boolean;
8014: Function PointInEllipse1( const P : TPoint; Left, Top, Right, Bottom : Integer ) : Boolean;
8015: Function DelphiToLocalFormat( const Format : String ) : String
8016: Function LocalToDelphiFormat( const Format : String ) : String
8017: Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
8018: Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8019: Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
  AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
8020: TTeeSortCompare', 'Function ( a, b : Integer ) : Integer
8021: TTeeSortSwap', 'Procedure ( a, b : Integer )
8022: Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8023: Function TeeGetUniqueName( AOwner : TComponent; const AStartTime : String ) : string
8024: Function TeeExtractField( St : String; Index : Integer ) : String;
8025: Function TeeExtractField1( St : String; Index : Integer; const Separator : String ) : String;
8026: Function TeeNumFields( St : String ) : Integer;
8027: Function TeeNumFields1( const St, Separator : String ) : Integer;
8028: Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap )

```

```

8029: Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
8030: // TColorArray', 'array of TColor
8031: Function GetDefaultColor( const Index : Integer) : TColor
8032: Procedure SetDefaultColorPalette;
8033: Procedure SetDefaultColorPalette( const Palette : array of TColor);
8034: 'TeeCheckBoxSize','LongInt'( 11);
8035: Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8036: Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
8037: Function TryStrToFloat( const S : String; var Value : Double) : Boolean
8038: Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
8039: Procedure TeeTranslateControl( AControl : TControl);
8040: Procedure TeeTranslateControl( AControl : TControl; const ExcludeChildren : array of TControl);
8041: Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
8042: //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8043: Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8044: //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8045: //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8046: Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8047: Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8048: Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8049: Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8050: Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8051: Procedure TeeSaveStringOption( const AKey, Value : String)
8052: Function TeeDefaultXMLEncoding : String
8053: Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8054: TeeWindowHandle', 'Integer
8055: Procedure TeeGoToURL( Handle : TeeWindowHandle; const URL : String)
8056: Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8057: Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8058: end;
8059:
8060:
8061: using mXBDEUtils
8062: ****
8063: Procedure SetAlias( aAlias, aDirectory : String)
8064: Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
Desired:Variant;Size:Byte);
8065: Function GetFileVersionNumber( const FileName : String) : TVersionNo
8066: Procedure SetBDE( aPath, aNode, aValue : String)
8067: function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8068: Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8069: Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8070: Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8071: Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8072: Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8073:
8074:
8075: procedure SIRegister_cDateTime(CL: TPPSPascalCompiler);
8076: begin
8077: AddClassN(FindClass('TOBJECT'),'EDateTime
8078: Function DatePart( const D : TDateTime) : Integer
8079: Function TimePart( const D : TDateTime) : Double
8080: Function Century( const D : TDateTime) : Word
8081: Function Year( const D : TDateTime) : Word
8082: Function Month( const D : TDateTime) : Word
8083: Function Day( const D : TDateTime) : Word
8084: Function Hour( const D : TDateTime) : Word
8085: Function Minute( const D : TDateTime) : Word
8086: Function Second( const D : TDateTime) : Word
8087: Function Millisecond( const D : TDateTime) : Word
8088: ('OneDay','Extended').setExtended( 1.0);
8089: ('OneHour','Extended').SetExtended( OneDay / 24);
8090: ('OneMinute','Extended').SetExtended( OneHour / 60);
8091: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8092: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8093: ('OneWeek','Extended').SetExtended( OneDay * 7);
8094: ('HoursPerDay','Extended').SetExtended( 24);
8095: ('MinutesPerHour','Extended').SetExtended( 60);
8096: ('SecondsPerMinute','Extended').SetExtended( 60);
8097: Procedure SetYear( var D : TDateTime; const Year : Word)
8098: Procedure SetMonth( var D : TDateTime; const Month : Word)
8099: Procedure SetDay( var D : TDateTime; const Day : Word)
8100: Procedure SetHour( var D : TDateTime; const Hour : Word)
8101: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8102: Procedure SetSecond( var D : TDateTime; const Second : Word)
8103: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8104: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8105: Function IsEqual( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8106: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8107: Function IsAM( const D : TDateTime) : Boolean
8108: Function IsPM( const D : TDateTime) : Boolean
8109: Function IsMidnight( const D : TDateTime) : Boolean
8110: Function IsNoon( const D : TDateTime) : Boolean
8111: Function IsSunday( const D : TDateTime) : Boolean
8112: Function IsMonday( const D : TDateTime) : Boolean
8113: Function IsTuesday( const D : TDateTime) : Boolean
8114: Function IsWednesday( const D : TDateTime) : Boolean
8115: Function IsThursday( const D : TDateTime) : Boolean
8116: Function IsFriday( const D : TDateTime) : Boolean

```

```

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

```

```

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

```

```

8294: Procedure SelfTestCTimer
8295: end;
8296:
8297: procedure SIRегистер_cRandom(CL: TPSPPascalCompiler);
8298: begin
8299:   Function RandomSeed : LongWord
8300:   Procedure AddEntropy( const Value : LongWord)
8301:   Function RandomUniform : LongWord;
8302:   Function RandomUniforml( const N : Integer) : Integer;
8303:   Function RandomBoolean : Boolean
8304:   Function RandomByte : Byte
8305:   Function RandomByteNonZero : Byte
8306:   Function RandomWord : Word
8307:   Function RandomInt64 : Int64;
8308:   Function RandomInt64l( const N : Int64) : Int64;
8309:   Function RandomHex( const Digits : Integer) : String
8310:   Function RandomFloat : Extended
8311:   Function RandomAlphaStr( const Length : Integer) : AnsiString
8312:   Function RandomPseudoWord( const Length : Integer) : AnsiString
8313:   Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8314:   Function mwcRandomLongWord : LongWord
8315:   Function urnRandomLongWord : LongWord
8316:   Function moaRandomFloat : Extended
8317:   Function mwcRandomFloat : Extended
8318:   Function RandomNormalF : Extended
8319:   Procedure SelfTestCRandom
8320: end;
8321:
8322: procedure SIRегистер_SynEditMiscProcs(CL: TPSPPascalCompiler);
8323: begin
8324:   // PIntArray', '^TIntArray // will not work
8325:   Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8326:   TConvertTabsProcEx, function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8327:   Function synMax( x, y : integer ) : integer
8328:   Function synMin( x, y : integer ) : integer
8329:   Function synMinMax( x, mi, ma : integer ) : integer
8330:   Procedure synSwapInt( var l, r : integer )
8331:   Function synMaxPoint( const P1, P2 : TPoint ) : TPoint
8332:   Function synMinPoint( const P1, P2 : TPoint ) : TPoint
8333:   //Function synGetIntArray( Count : Cardinal; initialValue : integer ) : PIntArray
8334:   Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect )
8335:   Function synGetBestConvertTabsProc( TabWidth : integer ) : TConvertTabsProc
8336:   Function synConvertTabs( const Line : AnsiString; TabWidth : integer ) : AnsiString
8337:   Function synGetBestConvertTabsProcEx( TabWidth : integer ) : TConvertTabsProcEx
8338:   Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8339:   Function synGetExpandedLength( const aStr : string; aTabWidth : integer ) : integer
8340:   Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string ) : integer
8341:   Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8342:   Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8343:   Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8344:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8345:   +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8346:   ('C3_NONSPACING','LongInt'( 1 );
8347:   'C3_DIACRITIC','LongInt'( 2 );
8348:   'C3_VOWELMARK','LongInt'( 4 );
8349:   ('C3_SYMBOL','LongInt'( 8 );
8350:   ('C3_KATAKANA','LongWord( $0010);
8351:   ('C3_HIRAGANA','LongWord( $0020);
8352:   ('C3_HALFWIDTH','LongWord( $0040);
8353:   ('C3_FULLWIDTH','LongWord( $0080);
8354:   ('C3_IDEOGRAPH','LongWord( $0100);
8355:   ('C3_KASHIDA','LongWord( $0200);
8356:   ('C3_LEXICAL','LongWord( $0400);
8357:   ('C3_ALPHA','LongWord( $8000);
8358:   ('C3_NOTAPPLICABLE','LongInt'( 0 );
8359:   Function synStrScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8360:   Function synStrScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8361:   Function synIsStringType( Value : Word ) : TStringType
8362:   Function synGetEOL( Line : PChar ) : PChar
8363:   Function synEncodeString( s : string ) : string
8364:   Function synDecodeString( s : string ) : string
8365:   Procedure synFreeAndNil( var Obj: TObject )
8366:   Procedure synAssert( Expr : Boolean )
8367:   Function synLastDelimiter( const Delimiters, S : string ) : Integer
8368:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8369:   TReplaceFlags', 'set of TReplaceFlag )
8370:   Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8371:   Function synGetRValue( RGBValue : TColor ) : byte
8372:   Function synGetGValue( RGBValue : TColor ) : byte
8373:   Function synGetBValue( RGBValue : TColor ) : byte
8374:   Function synRGB( r, g, b : Byte ) : Cardinal
8375:   // THighlighterAttriProc', 'Function( Highlighter : TSynCustomHighlighter
8376:   // +'lighter; Attr:TSynHighlighterAttributes;UniqueAttrName:string;Params array of Pointer):Boolean;
8377:   //Function synEnumHighlighterAttrs( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
8378:   HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer ) : Boolean
8379:   Function synCalcFCS( const ABuf, ABufSize : Cardinal ) : Word
8380:   Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
     AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8380: end;

```

```

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

```

```

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

```

```

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

```

```

8644: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B, A : Byte)
8645: Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8646: Function RedComponent( Color32 : TColor32) : Integer
8647: Function GreenComponent( Color32 : TColor32) : Integer
8648: Function BlueComponent( Color32 : TColor32) : Integer
8649: Function AlphaComponent( Color32 : TColor32) : Integer
8650: Function Intensity( Color32 : TColor32) : Integer
8651: Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8652: Function HSLtoRGB( H, S, L : Single) : TColor32;
8653: Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8654: Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8655: Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8656: Function WinPalette( const P : TPalette32) : HPALETTE
8657: Function FloatPoint( X, Y : Single) : TFloatPoint;
8658: Function FloatPoint1( const P : TPoint) : TFloatPoint;
8659: Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8660: Function FixedPoint( X, Y : Integer) : TFixedPoint;
8661: Function FixedPoint1( X, Y : Single) : TFixedPoint;
8662: Function FixedPoint2( const P : TPoint) : TFixedPoint;
8663: Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8664: AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )')
8665: Function MakeRect( const L, T, R, B : Integer) : TRect;
8666: Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8667: Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8668: Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8669: Function FixedRect1( const ARect : TRect) : TRect;
8670: Function FixedRect2( const FR : TFloatRect) : TRect;
8671: Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8672: Function FloatRect1( const ARect : TRect) : TFloatRect;
8673: Function FloatRect2( const FXR : TRect) : TFloatRect;
8674: Function GIIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8675: Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8676: Function GUUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8677: Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8678: Function GEEqualRect( const R1, R2 : TRect) : Boolean;
8679: Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8680: Procedure GIInflateRect( var R : TRect; Dx, Dy : Integer);
8681: Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8682: Procedure GOFFsetRect( var R : TRect; Dx, Dy : Integer);
8683: Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8684: Function IsRectEmpty( const R : TRect) : Boolean;
8685: Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8686: Function GPTInRect( const R : TRect; const P : TPoint) : Boolean;
8687: Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8688: Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8689: Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8690: Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8691: Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8692: Function MessageBeep( uType : UINT) : BOOL
8693: Function ShowCursor( bShow : BOOL) : Integer
8694: Function SetCursorPos( X, Y : Integer) : BOOL
8695: Function SetCursor( hCursor : HICON) : HCURSOR
8696: Function GetCursorPos( var lpPoint : TPoint) : BOOL
8697: //Function ClipCursor( lpRect : PRect) : BOOL
8698: Function GetClipCursor( var lpRect : TRect) : BOOL
8699: Function GetCursor : HCURSOR
8700: Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8701: Function GetCaretBlinkTime : UINT
8702: Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8703: Function DestroyCaret : BOOL
8704: Function HideCaret( hWnd : HWND) : BOOL
8705: Function ShowCaret( hWnd : HWND) : BOOL
8706: Function SetCaretPos( X, Y : Integer) : BOOL
8707: Function GetCaretPos( var lpPoint : TPoint) : BOOL
8708: Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8709: Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8710: Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8711: Function WindowFromPoint( Point : TPoint) : HWND
8712: Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8713:
8714:
8715: procedure SIRegister_GR32_Math(CL: TPPSPascalCompiler);
8716: begin
8717:   Function FixedFloor( A : TFixed) : Integer
8718:   Function FixedCeil( A : TFixed) : Integer
8719:   Function FixedMul( A, B : TFixed) : TFixed
8720:   Function FixedDiv( A, B : TFixed) : TFixed
8721:   Function OneOver( Value : TFixed) : TFixed
8722:   Function FixedRound( A : TFixed) : Integer
8723:   Function FixedSqr( Value : TFixed) : TFixed
8724:   Function FixedSqrtLP( Value : TFixed) : TFixed
8725:   Function FixedSqrtHP( Value : TFixed) : TFixed
8726:   Function FixedCombine( W, X, Y : TFixed) : TFixed
8727:   Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8728:   Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8729:   Function GRHypot( const X, Y : TFloat) : TFloat;
8730:   Function Hypot1( const X, Y : Integer) : Integer;
8731:   Function FastSqrt( const Value : TFloat) : TFloat
8732:   Function FastSqrtBab1( const Value : TFloat) : TFloat

```

```

8733: Function FastSqrtBab2( const Value : TFloat ) : TFloat
8734: Function FastInvSqrt( const Value : Single ) : Single;
8735: Function MulDiv( Multiplicand, Multiplier, Divisor : Integer ) : Integer
8736: Function GRIsPowerOf2( Value : Integer ) : Boolean
8737: Function PrevPowerOf2( Value : Integer ) : Integer
8738: Function NextPowerOf2( Value : Integer ) : Integer
8739: Function Average( A, B : Integer ) : Integer
8740: Function GRSign( Value : Integer ) : Integer
8741: Function FloatMod( x, y : Double ) : Double
8742: end;
8743:
8744: procedure SIRegister_GR32_LowLevel(CL: TPSCompiler);
8745: begin
8746:   Function Clamp( const Value : Integer ) : Integer;
8747:   Procedure GRFillWord( var X, Count : Cardinal; Value : Longword )
8748:   Function StackAlloc( Size : Integer ) : Pointer
8749:   Procedure StackFree( P : Pointer )
8750:   Procedure Swap( var A, B : Pointer );
8751:   Procedure Swap1( var A, B : Integer );
8752:   Procedure Swap2( var A, B : TFixed );
8753:   Procedure Swap3( var A, B : TColor32 );
8754:   Procedure TestSwap( var A, B : Integer );
8755:   Procedure TestSwap1( var A, B : TFixed );
8756:   Function TestClip( var A, B : Integer; const Size : Integer ) : Boolean;
8757:   Function TestClip1( var A, B : Integer; const Start, Stop : Integer ) : Boolean;
8758:   Function GRConstrain( const Value, Lo, Hi : Integer ) : Integer;
8759:   Function Constrain1( const Value, Lo, Hi : Single ) : Single;
8760:   Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer ) : Integer
8761:   Function GRMin( const A, B, C : Integer ) : Integer;
8762:   Function GRMax( const A, B, C : Integer ) : Integer;
8763:   Function Clamp( Value, Max : Integer ) : Integer;
8764:   Function Clamp1( Value, Min, Max : Integer ) : Integer;
8765:   Function Wrap( Value, Max : Integer ) : Integer;
8766:   Function Wrap1( Value, Min, Max : Integer ) : Integer;
8767:   Function Wrap3( Value, Max : Single ) : Single;;
8768:   Function WrapPow2( Value, Max : Integer ) : Integer;
8769:   Function WrapPow21( Value, Min, Max : Integer ) : Integer;
8770:   Function Mirror( Value, Max : Integer ) : Integer;
8771:   Function Mirror1( Value, Min, Max : Integer ) : Integer;
8772:   Function MirrorPow2( Value, Max : Integer ) : Integer;
8773:   Function MirrorPow21( Value, Min, Max : Integer ) : Integer;
8774:   Function GetOptimalWrap( Max : Integer ) : TWrapProc;
8775:   Function GetOptimalWrap1( Min, Max : Integer ) : TWrapProcEx;
8776:   Function GetOptimalMirror( Max : Integer ) : TWrapProc;
8777:   Function GetOptimalMirror1( Min, Max : Integer ) : TWrapProcEx;
8778:   Function GetWrapProc( WrapMode : TWrapMode ) : TWrapProc;
8779:   Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer ) : TWrapProc;
8780:   Function GetWrapProcEx( WrapMode : TWrapMode ) : TWrapProcEx;
8781:   Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer ) : TWrapProcEx;
8782:   Function Div255( Value : Cardinal ) : Cardinal;
8783:   Function SAR_4( Value : Integer ) : Integer;
8784:   Function SAR_8( Value : Integer ) : Integer;
8785:   Function SAR_9( Value : Integer ) : Integer;
8786:   Function SAR_11( Value : Integer ) : Integer;
8787:   Function SAR_12( Value : Integer ) : Integer;
8788:   Function SAR_13( Value : Integer ) : Integer;
8789:   Function SAR_14( Value : Integer ) : Integer;
8790:   Function SAR_15( Value : Integer ) : Integer;
8791:   Function SAR_16( Value : Integer ) : Integer;
8792:   Function ColorSwap( WinColor : TColor ) : TColor32;
8793: end;
8794:
8795: procedure SIRegister_GR32_Filters(CL: TPSCompiler);
8796: begin
8797:   AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )');
8798:   Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components );
8799:   Procedure CopyComponents1(Dst:TCustomBitmap32;DstX,DstY:Int32;Src:TCustomBitmap32;SrcRect:TRect;Components:TColor32Comp);
8800:   Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32 );
8801:   Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean );
8802:   Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32 );
8803:   Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components );
8804:   Procedure InvertRGB( Dst, Src : TCustomBitmap32 );
8805:   Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean );
8806:   Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32 );
8807:   Function CreateBitmask( Components : TColor32Components ) : TColor32;
8808:   Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect; Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8809:   Procedure ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8810:   Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean );
8811: end;
8812:
8813:
8814: procedure SIRegister_JclNTFS(CL: TPSCompiler);
8815: begin
8816:   AddClassN(FindClass('TOBJECT'), 'EJclNtfsError');
8817:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
8818:   Function NtfsGetCompression( const FileName : string; var State : Short ) : Boolean;

```

```

8819: Function NtfsGetCompression1( const FileName : string ) : TFileCompressionState;
8820: Function NtfsSetCompression( const FileName : string; const State : Short ) : Boolean
8821: Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState )
8822: Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8823: Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8824: Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8825: //AddTypeS('TNTfsAllocRanges', 'record Entries : Integer; Data : PFileAlloc'
8826: //+'+tedRangeBuffer; MoreData : Boolean; end
8827: Function NtfsSetSparse( const FileName : string ) : Boolean
8828: Function NtfsZeroDataByHandle( const Handle: THandle; const First, Last : Int64 ) : Boolean
8829: Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64 ) : Boolean
8830: //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
Ranges:TNTfsAllocRanges):Boolean;
8831: //Function NtfsGetAllocRangeEntry( const Ranges : TNTfsAllocRanges;
Index:Integer):TFileAllocatedRangeBuffer
8832: Function NtfsSparseStreamsSupported( const Volume : string ) : Boolean
8833: Function NtfsGetSparse( const FileName : string ) : Boolean
8834: Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD ) : Boolean
8835: Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword ) : Boolean
8836: //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8837: Function NtfsGetReparseTag( const Path : string; var Tag : DWORD ) : Boolean
8838: Function NtfsReparsePointsSupported( const Volume : string ) : Boolean
8839: Function NtfsFileHasReparsePoint( const Path : string ) : Boolean
8840: Function NtfsIsFolderMountPoint( const Path : string ) : Boolean
8841: Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char ) : Boolean
8842: Function NtfsMountVolume( const Volume : Char; const MountPoint : string ) : Boolean
8843: AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter ')
8844: Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8845: Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8846: Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8847: Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8848: Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped ) : Boolean
8849: Function NtfsCreateJunctionPoint( const Source, Destination : string ) : Boolean
8850: Function NtfsDeleteJunctionPoint( const Source : string ) : Boolean
8851: Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string ) : Boolean
8852: AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec '
+ 'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8853: AddTypeS('TStreamIds', 'set of TStreamId
8854: AddTypes('TInternalFindStreamData', 'record FileHandle : THandle; Context '
+ ': __Pointer; StreamIds : TStreamIds; end
8855: AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
+ 'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8856: Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8857: Function NtfsFindNextStream( var Data : TFindStreamData ) : Boolean
8858: Function NtfsFindStreamClose( var Data : TFindStreamData ) : Boolean
8859: Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string ) : Boolean
8860: AddTypeS('TNTfsHardLinkInfo', 'record LinkCount : Cardinal; FileInfo : Int64; end
8861: Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo ) : Boolean
8862: Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
List:TStrings):Bool;
8863: Function NtfsDeleteHardLinks( const FileName : string ) : Boolean
8864: Function JclAppInstances : TJclAppInstances;
8865: Function JclAppInstances1( const UniqueAppIdGuidStr : string ) : TJclAppInstances;
8866: Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND ) : TJclAppInstDataKind
8867: Procedure ReadMessageData( const Message : TMessage; var Data : __Pointer; var Size : Integer )
8868: Procedure ReadMessageString( const Message : TMessage; var S : string )
8869: Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings )
8870:
8871:
8872:
8873:
8874:
8875: (*-----*)
8876: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8877: begin
8878:   FindClass('TOBJECT','EJclGraphicsError
8879:   TDynIntegerArrayArray', 'array of TDynIntegerArray
8880:   TDynPointArray', 'array of TPoint
8881:   TDynDynPointArrayArray', 'array of TDynPointArray
8882:   TPointF', 'record X : Single; Y : Single; end
8883:   TDynPointArrayF', 'array of TPointF
8884:   TDrawMode2', '( dmOpaque, dmBlend )
8885:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8886:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8887:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8888:   TMatrix3d', 'record array[0..2,0..2] of extended end
8889:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8890:   TScanLine', 'array of Integer
8891:   TScanLines', 'array of TScanLine
8892:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8893:   TGradientDirection', '( gdVertical, gdHorizontal )
8894:   TPolyFillMode', '( fmAlternate, fmWinding )
8895:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8896:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8897:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8898:   SIRegister_TJclDesktopCanvas(CL);
8899:   FindClass('TOBJECT','TJclRegion
8900:   SIRegister_TJclRegionInfo(CL);
8901:   SIRegister_TJclRegion(CL);
8902:   SIRegister_TJclThreadPersistent(CL);
8903:   SIRegister_TJclCustomMap(CL);
8904:   SIRegister_TJclBitmap32(CL);

```

```

8905:  SIRегистер_TJclByteMap(CL);
8906:  SIRегистер_TJclTransformation(CL);
8907:  SIRегистер_TJclLinearTransformation(CL);
8908:  Procedure Stretch(NewWidth,
8909:    NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8910:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitmap; X, Y, Width, Height : Integer)
8911:  Function GetAntialiasedBitmap( const Bitmap : TBitmap ) : TBitmap
8912:  Procedure BitmapToJpeg( const FileName : string )
8913:  Procedure JpegToBitmap( const FileName : string )
8914:  Function ExtractIconCount( const FileName : string ) : Integer
8915:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer ) : HICON
8916:  Function IconToBitmapJ( Icon : HICON ) : HBITMAP
8917:  Procedure
8918:    BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8919:  Procedure StretchTransfer(Dst:TJclBitmap32;
8920:    DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8921:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8922:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect )
8923:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection : TGradientDirection) : Boolean;
8924:  Function CreateRegionFromBitmap(Bitmap : TBitmap; RegionColor:TColor;
8925:    RegionBitmapMode:TJclRegionBitmapMode): HGRN
8926:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND );
8927:  Procedure ScreenShot1( bm : TBitmap );
8928:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8929:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8930:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8931:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8932:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8933:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32);
8934:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32 )
8935:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32 )
8936:  Procedure Invert( Dst, Src : TJclBitmap32 )
8937:  Procedure InvertRGB( Dst, Src : TJclBitmap32 )
8938:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32 )
8939:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8 )
8940:  Procedure SetGamma( Gamma : Single )
8941: end;
8942:
8943: (*-----*)
8944: procedure SIRегистер_JclSynch(CL: TPSPPascalCompiler);
8945: begin
8946:  Function LockedAdd( var Target : Integer; Value : Integer ) : Integer
8947:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer ) : Integer;
8948:  Function LockedCompareExchange1( var Target : __Pointer; Exch, Comp : __Pointer ) : Pointer;
8949:  Function LockedDec( var Target : Integer ) : Integer
8950:  Function LockedExchange( var Target : Integer; Value : Integer ) : Integer
8951:  Function LockedExchangeAdd( var Target : Integer; Value : Integer ) : Integer
8952:  Function LockedExchangeDec( var Target : Integer ) : Integer
8953:  Function LockedExchangeInc( var Target : Integer ) : Integer
8954:  Function LockedExchangeSub( var Target : Integer; Value : Integer ) : Integer
8955:  Function LockedInc( var Target : Integer ) : Integer
8956:  Function LockedSub( var Target : Integer; Value : Integer ) : Integer
8957:  TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )'
8958:  SIRегистер_TJclDispatcherObject(CL);
8959:  Function WaitForMultipleObjects(const Objects:array of
8960:    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:  POptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8971:  TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8972:  TMrewThreadInfo', 'record Threadid: Longword; RecursionCount: Integer; Reader : Boolean; end
8973:  TMrewThreadInfoArray', 'array of TMrewThreadInfo
8974:  SIRегистер_TJclMultiReadExclusiveWrite(CL);
8975:  PMetSectSharedInfo', '^PMetSectSharedInfo // will not work
8976:  PMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8977:  +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8978:  PMeteredSection', '^PMeteredSection // will not work
8979:  TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8980:  SIRегистер_TJclMeteredSection(CL);
8981:  EEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8982:  TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8983:  TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8984:  TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8985:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection ) : Boolean
8986:  Function QueryEvent( Handle : THandle; var Info : TEventInfo ) : Boolean
8987:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo ) : Boolean

```

```

8987: Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts ) : Boolean
8988: Function QueryTimer( Handle : THandle; var Info : TTimerInfo ) : Boolean
8989: FindClass('TOBJECT'), 'EJclWin32HandleObjectError
8990: FindClass('TOBJECT'), 'EJclDispatcherObjectError
8991: FindClass('TOBJECT'), 'EJclCriticalSectionError
8992: FindClass('TOBJECT'), 'EJclEventError
8993: FindClass('TOBJECT'), 'EJclWaitableTimerError
8994: FindClass('TOBJECT'), 'EJclSemaphoreError
8995: FindClass('TOBJECT'), 'EJclMutexError
8996: FindClass('TOBJECT'), 'EJclMeteredSectionError
8997: end;
8998:
8999:
9000: //*****unit uPSI_mORMotReport;
9001: Procedure SetCurrentPrinterAsDefault
9002: Function CurrentPrinterName : string
9003: Function mCurrentPrinterPaperSize : string
9004: Procedure UseDefaultPrinter
9005:
9006: procedure SIRegisterTSTREAM(C1: TPSCompiler);
9007: begin
9008:   with FindClass('TOBJECT'), 'TStream') do begin
9009:     IsAbstract := True;
9010:     //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
9011:     // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
9012:     function Read(Buffer:String;Count:LongInt):LongInt
9013:     function Write(Buffer:String;Count:LongInt):LongInt
9014:     function ReadString(Buffer:String;Count:LongInt):LongInt    //FileStream
9015:     function WriteString(Buffer:String;Count:LongInt):LongInt
9016:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
9017:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
9018:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt';
9019:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt';
9020:
9021:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
9022:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
9023:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
9024:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
9025:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
9026:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
9027:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
9028:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
9029:
9030:     function Seek(Offset:LongInt;Origin:Word):LongInt
9031:     procedure ReadBuffer(Buffer:String;Count:LongInt)
9032:     procedure WriteBuffer(Buffer:String;Count:LongInt)
9033:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt');
9034:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt');
9035:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt');
9036:     procedure WriteBufferFloat(Buffer:Double;Count:LongInt');
9037:
9038:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
9039:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9040:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9041:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9042:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9043:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9044:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9045:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9046:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9047:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9048:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9049:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9050:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9051:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9052:
9053:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
9054:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
9055:     procedure ReadBufferO(Buffer: TObject;Count:LongInt');
9056:     procedure WriteBufferO(Buffer: TObject;Count:LongInt');
9057:     //READBUFFERAC
9058:     function InstanceSize: Longint
9059:     Procedure FixupResourceHeader( FixupInfo : Integer )
9060:     Procedure ReadResHeader
9061:
9062: {$IFDEF DELPHI4UP}
9063:     function CopyFrom(Source:TStream;Count:Int64):LongInt
9064:     {$ELSE}
9065:     function CopyFrom(Source:TStream;Count:Integer):LongInt
9066:     {$ENDIF}
9067:     RegisterProperty('Position', 'LongInt', iptrw);
9068:     RegisterProperty('Size', 'LongInt', iptrw);
9069:   end;
9070: end;
9071:
9072:
9073: { ****
9074: Unit DMATH - Interface for DMATH.DLL
9075: **** }

```

```

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

```

```

9165: { 10^X }
9166: function Log(X : Float) : Float; external 'dmath';
9167: { Natural log }
9168: function Log2(X : Float) : Float; external 'dmath';
9169: { Log, base 2 }
9170: function Log10(X : Float) : Float; external 'dmath';
9171: { Decimal log }
9172: function LogA(X, A : Float) : Float; external 'dmath';
9173: { Log, base A }
9174: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9175: { X^N }
9176: function Power(X, Y : Float) : Float; external 'dmath';
9177: { X^Y, X >= 0 }
9178: { -----
9179:   Trigonometric functions
9180: ----- }
9181: function Pythag(X, Y : Float) : Float; external 'dmath';
9182: { Sqrt(X^2 + Y^2) }
9183: function FixAngle(Theta : Float) : Float; external 'dmath';
9184: { Set Theta in -Pi..Pi }
9185: function Tan(X : Float) : Float; external 'dmath';
9186: { Tangent }
9187: function ArcSin(X : Float) : Float; external 'dmath';
9188: { Arc sinus }
9189: function ArcCos(X : Float) : Float; external 'dmath';
9190: { Arc cosinus }
9191: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9192: { Angle (Ox, OM) with M(X,Y) }
9193: { -----
9194:   Hyperbolic functions
9195: ----- }
9196: function Sinh(X : Float) : Float; external 'dmath';
9197: { Hyperbolic sine }
9198: function Cosh(X : Float) : Float; external 'dmath';
9199: { Hyperbolic cosine }
9200: function Tanh(X : Float) : Float; external 'dmath';
9201: { Hyperbolic tangent }
9202: function ArcSinh(X : Float) : Float; external 'dmath';
9203: { Inverse hyperbolic sine }
9204: function ArcCosh(X : Float) : Float; external 'dmath';
9205: { Inverse hyperbolic cosine }
9206: function ArcTanh(X : Float) : Float; external 'dmath';
9207: { Inverse hyperbolic tangent }
9208: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9209: { Sinh & Cosh }
9210: { -----
9211:   Gamma function and related functions
9212: ----- }
9213: function Fact(N : Integer) : Float; external 'dmath';
9214: { Factorial }
9215: function SgnGamma(X : Float) : Integer; external 'dmath';
9216: { Sign of Gamma function }
9217: function Gamma(X : Float) : Float; external 'dmath';
9218: { Gamma function }
9219: function LnGamma(X : Float) : Float; external 'dmath';
9220: { Logarithm of Gamma function }
9221: function Stirling(X : Float) : Float; external 'dmath';
9222: { Stirling's formula for the Gamma function }
9223: function StirLog(X : Float) : Float; external 'dmath';
9224: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9225: function DiGamma(X : Float) : Float; external 'dmath';
9226: { Digamma function }
9227: function TriGamma(X : Float) : Float; external 'dmath';
9228: { Trigamma function }
9229: function IGamma(A, X : Float) : Float; external 'dmath';
9230: { Incomplete Gamma function }
9231: function JGamma(A, X : Float) : Float; external 'dmath';
9232: { Complement of incomplete Gamma function }
9233: function InvGamma(A, P : Float) : Float; external 'dmath';
9234: { Inverse of incomplete Gamma function }
9235: function Erf(X : Float) : Float; external 'dmath';
9236: { Error function }
9237: function Erfc(X : Float) : Float; external 'dmath';
9238: { Complement of error function }
9239: { -----
9240:   Beta function and related functions
9241: ----- }
9242: function Beta(X, Y : Float) : Float; external 'dmath';
9243: { Beta function }
9244: function IBeta(A, B, X : Float) : Float; external 'dmath';
9245: { Incomplete Beta function }
9246: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9247: { Inverse of incomplete Beta function }
9248: { -----
9249:   Lambert's function
9250: ----- }
9251: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9252: { -----
9253:   Binomial distribution

```

```

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

```

```

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

```

```

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

```

```

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

```

```

9610: { 32-bit random real in (0,1) }
9611: function RanGen53 : Float; external 'dmath';
9612: { 53-bit random real in [0,1) }
9613: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9614: { Initializes the 'Multiply with carry' random number generator }
9615: function IRanMWC : RNG_IntType; external 'dmath';
9616: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9617: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9618: { Initializes Mersenne Twister generator with a seed }
9619: procedure InitMTbyArray(InitKey : array of RNG_LongType;
9620:                           KeyLength : Word); external 'dmath';
9621: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9622: function IRanMT : RNG_IntType; external 'dmath';
9623: { Random integer from MT generator }
9624: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9625: { Initializes the UVAG generator with a string }
9626: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9627: { Initializes the UVAG generator with an integer }
9628: function IRanUVAG : RNG_IntType; external 'dmath';
9629: { Random integer from UVAG generator }
9630: function RanGaussStd : Float; external 'dmath';
9631: { Random number from standard normal distribution }
9632: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9633: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9634: procedure RanMult(M : TVector; L : TMatrix;
9635:                      Lb, Ub : Integer;
9636:                      X : TVector); external 'dmath';
9637: { Random vector from multinormal distribution (correlated) }
9638: procedure RanMultIndep(M, S : TVector;
9639:                          Lb, Ub : Integer;
9640:                          X : TVector); external 'dmath';
9641: { Random vector from multinormal distribution (uncorrelated) }
9642: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9643: { Initializes Metropolis-Hastings parameters }
9644: procedure GetMHParams(var NCycles, MaxSim, SavedSim:Integer); external 'dmath';
9645: { Returns Metropolis-Hastings parameters }
9646: procedure Hastings(Func : TFuncNVar;
9647:                        T : Float;
9648:                        X : TVector;
9649:                        V : TMatrix;
9650:                        Lb, Ub : Integer;
9651:                        Xmat : TMatrix;
9652:                        X_min : TVector;
9653:                        var F_min : Float); external 'dmath';
9654: { Simulation of a probability density function by Metropolis-Hastings }
9655: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9656: { Initializes Simulated Annealing parameters }
9657: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9658: { Initializes log file }
9659: procedure SimAnn(Func : TFuncNVar;
9660:                      X, Xmin, Xmax : TVector;
9661:                      Lb, Ub : Integer;
9662:                      var F_min : Float); external 'dmath';
9663: { Minimization of a function of several var. by simulated annealing }
9664: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9665: { Initializes Genetic Algorithm parameters }
9666: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9667: { Initializes log file }
9668: procedure GenAlg(Func : TFuncNVar;
9669:                      X, Xmin, Xmax : TVector;
9670:                      Lb, Ub : Integer;
9671:                      var F_min : Float); external 'dmath';
9672: { Minimization of a function of several var. by genetic algorithm }
9673: { -----
9674:   Statistics
9675:   ----- }
9676: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9677: { Mean of sample X }
9678: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9679: { Minimum of sample X }
9680: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9681: { Maximum of sample X }
9682: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9683: { Median of sample X }
9684: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9685: { Standard deviation estimated from sample X }
9686: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9687: { Standard deviation of population }
9688: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9689: { Correlation coefficient }
9690: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9691: { Skewness of sample X }
9692: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9693: { Kurtosis of sample X }
9694: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9695: { Quick sort (ascending order) }
9696: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9697: { Quick sort (descending order) }
9698: procedure Interval(x1, x2 : Float);

```

```

9699:           MinDiv, MaxDiv      : Integer;
9700:           var Min, Max, Step : Float); external 'dmath';
9701: { Determines an interval for a set of values }
9702: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9703:                       var XMin, XMax, XStep : Float); external 'dmath';
9704: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9705: procedure StudIndep(N1, N2      : Integer;
9706:                       M1, M2, S1, S2 : Float;
9707:                       var T      : Float;
9708:                       var DoF      : Integer); external 'dmath';
9709: { Student t-test for independent samples }
9710: procedure StudPaired(X, Y      : TVector;
9711:                         Lb, Ub : Integer;
9712:                         var T      : Float;
9713:                         var DoF : Integer); external 'dmath';
9714: { Student t-test for paired samples }
9715: procedure AnOVal(Ns      : Integer;
9716:                     N      : TIntVector;
9717:                     M, S      : TVector;
9718:                     var V_f, V_r, F : Float;
9719:                     var DoF_f, DoF_r : Integer); external 'dmath';
9720: { One-way analysis of variance }
9721: procedure AnOva2(NA, NB, Nobs : Integer;
9722:                     M, S      : TMatrix;
9723:                     V, F      : TVector;
9724:                     DoF      : TIntVector); external 'dmath';
9725: { Two-way analysis of variance }
9726: procedure Snedecor(N1, N2      : Integer;
9727:                       S1, S2      : Float;
9728:                       var F      : Float;
9729:                       var DoF1, DoF2 : Integer); external 'dmath';
9730: { Snedecor's F-test (comparison of two variances) }
9731: procedure Bartlett(Ns      : Integer;
9732:                       N      : TIntVector;
9733:                       S      : TVector;
9734:                       var Khi2 : Float;
9735:                       var DoF : Integer); external 'dmath';
9736: { Bartlett's test (comparison of several variances) }
9737: procedure Mann_Whitney(N1, N2      : Integer;
9738:                           X1, X2      : TVector;
9739:                           var U, Eps : Float); external 'dmath';
9740: { Mann-Whitney test }
9741: procedure Wilcoxon(X, Y      : TVector;
9742:                         Lb, Ub : Integer;
9743:                         var Ndiff : Integer;
9744:                         var T, Eps : Float); external 'dmath';
9745: { Wilcoxon test }
9746: procedure Kruskal_Wallis(Ns      : Integer;
9747:                             N      : TIntVector;
9748:                             X      : TMatrix;
9749:                             var H : Float;
9750:                             var DoF : Integer); external 'dmath';
9751: { Kruskal-Wallis test }
9752: procedure Khi2_Conform(N_cls   : Integer;
9753:                           N_estim : Integer;
9754:                           Obs     : TIntVector;
9755:                           Calc    : TVector;
9756:                           var Khi2 : Float;
9757:                           var DoF : Integer); external 'dmath';
9758: { Khi-2 test for conformity }
9759: procedure Khi2_Indep(N_lin   : Integer;
9760:                           N_col  : Integer;
9761:                           Obs    : TIntMatrix;
9762:                           var Khi2 : Float;
9763:                           var DoF : Integer); external 'dmath';
9764: { Khi-2 test for independence }
9765: procedure Woolf_Conform(N_cls   : Integer;
9766:                           N_estim : Integer;
9767:                           Obs     : TIntVector;
9768:                           Calc    : TVector;
9769:                           var G : Float;
9770:                           var DoF : Integer); external 'dmath';
9771: { Woolf's test for conformity }
9772: procedure Woolf_Indep(N_lin   : Integer;
9773:                           N_col  : Integer;
9774:                           Obs    : TIntMatrix;
9775:                           var G : Float;
9776:                           var DoF : Integer); external 'dmath';
9777: { Woolf's test for independence }
9778: procedure DimStatClassVector(var C : TStatClassVector;
9779:                               Ub : Integer); external 'dmath';
9780: { Allocates an array of statistical classes: C[0..Ub] }
9781: procedure Distrib(X      : TVector;
9782:                       Lb, Ub : Integer;
9783:                       A, B, H : Float;
9784:                       C      : TStatClassVector); external 'dmath';
9785: { Distributes an array X[Lb..Ub] into statistical classes }
9786: { -----
9787:  Linear / polynomial regression

```

```

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

```

```

9877: ----- }
9878: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9879: { Sets the optimization algorithm for nonlinear regression }
9880: function GetOptAlgo : TOptAlgo; external 'dmath';
9881: { Returns the optimization algorithm }
9882: procedure SetMaxParam(N : Byte); external 'dmath';
9883: { Sets the maximum number of regression parameters for nonlinear regression }
9884: function GetMaxParam : Byte; external 'dmath';
9885: { Returns the maximum number of regression parameters for nonlinear regression }
9886: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9887: { Sets the bounds on the I-th regression parameter }
9888: procedure GetParamBounds(I : Byte; var ParamMin, ParamMax:Float); external 'dmath';
9889: { Returns the bounds on the I-th regression parameter }
9890: procedure NLFit(RegFunc : TRegFunc;
9891:           DerivProc : TDervProc;
9892:           X, Y : TVector;
9893:           Lb, Ub : Integer;
9894:           MaxIter : Integer;
9895:           Tol : Float;
9896:           B : TVector;
9897:           FirstPar,
9898:           LastPar : Integer;
9899:           V : TMatrix); external 'dmath';
9900: { Unweighted nonlinear regression }
9901: procedure WNLFit(RegFunc : TRegFunc;
9902:           DerivProc : TDervProc;
9903:           X, Y, S : TVector;
9904:           Lb, Ub : Integer;
9905:           MaxIter : Integer;
9906:           Tol : Float;
9907:           B : TVector;
9908:           FirstPar,
9909:           LastPar : Integer;
9910:           V : TMatrix); external 'dmath';
9911: { Weighted nonlinear regression }
9912: procedure SetMCFile(FileName : String); external 'dmath';
9913: { Set file for saving MCMC simulations }
9914: procedure SimFit(RegFunc : TRegFunc;
9915:           X, Y : TVector;
9916:           Lb, Ub : Integer;
9917:           B : TVector;
9918:           FirstPar,
9919:           LastPar : Integer;
9920:           V : TMatrix); external 'dmath';
9921: { Simulation of unweighted nonlinear regression by MCMC }
9922: procedure WSimFit(RegFunc : TRegFunc;
9923:           X, Y, S : TVector;
9924:           Lb, Ub : Integer;
9925:           B : TVector;
9926:           FirstPar,
9927:           LastPar : Integer;
9928:           V : TMatrix); external 'dmath';
9929: { Simulation of weighted nonlinear regression by MCMC }
9930: { -----
9931: Nonlinear regression models
9932: ----- }

9933: procedure FracFit(X, Y : TVector;
9934:           Lb, Ub : Integer;
9935:           Deg1, Deg2 : Integer;
9936:           ConsTerm : Boolean;
9937:           MaxIter : Integer;
9938:           Tol : Float;
9939:           B : TVector;
9940:           V : TMatrix); external 'dmath';
9941: { Unweighted fit of rational fraction }
9942: procedure WFracFit(X, Y, S : TVector;
9943:           Lb, Ub : Integer;
9944:           Deg1, Deg2 : Integer;
9945:           ConsTerm : Boolean;
9946:           MaxIter : Integer;
9947:           Tol : Float;
9948:           B : TVector;
9949:           V : TMatrix); external 'dmath';
9950: { Weighted fit of rational fraction }

9951: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9952: { Returns the value of the rational fraction at point X }
9953: procedure ExpFit(X, Y : TVector;
9954:           Lb, Ub, Nexp : Integer;
9955:           ConsTerm : Boolean;
9956:           MaxIter : Integer;
9957:           Tol : Float;
9958:           B : TVector;
9959:           V : TMatrix); external 'dmath';
9960: { Unweighted fit of sum of exponentials }
9961: procedure WExpFit(X, Y, S : TVector;
9962:           Lb, Ub, Nexp : Integer;
9963:           ConsTerm : Boolean;
9964:           MaxIter : Integer;
9965:
```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

10411: end;
10412:
10413: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10414: begin
10415:   Procedure ClearModules
10416:   Procedure ReadMapFile( Fname : string )
10417:   Function AddressInfo( Address : dword ) : string
10418:   end;
10419:
10420: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10421: begin
10422:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOwner'
10423:   +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWrite'
10424:   +'teByOther, tpExecuteByOther )
10425:   TTarPermissions', 'set of TTarPermission
10426:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10427:   +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10428:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10429:   TTarModes', 'set of TTarMode
10430:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDate'
10431:   +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10432:   +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING; '
10433:   +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10434:   +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10435:   SIRegister_TTarArchive(CL);
10436:   SIRegister_TTarWriter(CL);
10437:   Function PermissionString( Permissions : TTarPermissions ) : STRING
10438:   Function ConvertFilename( Filename : STRING ) : STRING
10439:   Function FileTimeGMT( FileName : STRING ) : TDateTime;
10440:   Function FileTimeGMT1( SearchRec : TSearchRec ) : TDateTime;
10441:   Procedure ClearDirRec( var DirRec : TTarDirRec )
10442: end;
10443:
10444:
10445: //*****unit uPSI_TlHelp32;
10446: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10447: begin
10448:   Const('MAX_MODULE_NAME32','LongInt'( 255 );
10449:   Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD ) : THandle
10450:   Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001 );
10451:   Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002 );
10452:   Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004 );
10453:   Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008 );
10454:   Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000 );
10455:   tagHEAPLIST32', 'record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10456:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10457:   AddTypeS('THeapList32', 'tagHEAPLIST32
10458:   Const('HF32_DEFAULT','LongInt'( 1 );
10459:   Const('HF32_SHARED','LongInt'( 2 );
10460:   Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10461:   Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10462:   AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THHandle; dwAdr'
10463:   +'ress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10464:   +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10465:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10466:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10467:   Const('LF32_FIXED','LongWord').SetUInt( $00000001 );
10468:   Const('LF32_FREE','LongWord').SetUInt( $00000002 );
10469:   Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004 );
10470:   Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD ) : BOOL
10471:   Function Heap32Next( var lphe : THeapEntry32 ) : BOOL
10472:   DWORD; var lpNumberOfBytesRead : DWORD ) : BOOL
10473:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10474:   +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10475:   +'aPri : Longint; dwFlags : DWORD; end
10476:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10477:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10478:   Function Thread32First( hSnapshot : THHandle; var lppte : TThreadEntry32 ) : BOOL
10479:   Function Thread32Next( hSnapshot : THHandle; var lppte : TThreadEntry32 ) : BOOL
10480: end;
10481: Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042 );
10482: Const('EW_REBOOTSYSTEM','LongWord( $0043 );
10483: Const('EW_EXITANDEXECAPP','LongWord( $0044 );
10484: Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ) );
10485: Const('EWX_LOGOFF','LongInt'( 0 );
10486: Const('EWX_SHUTDOWN','LongInt'( 1 );
10487: Const('EWX_REBOOT','LongInt'( 2 );
10488: Const('EWX_FORCE','LongInt'( 4 );
10489: Const('EWX_POWEROFF','LongInt'( 8 );
10490: Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10 );
10491: Function GET_APPCOMMAND_LPARAM( const lParam : LongInt ) : Shortint
10492: Function GET_DEVICE_LPARAM( const lParam : LongInt ) : Word
10493: Function GET_MOUSEKEY_LPARAM( const lParam : LongInt ) : Word
10494: Function GET_FLAGS_LPARAM( const lParam : LongInt ) : Word
10495: Function GET_KEYSTATE_LPARAM( const lParam : LongInt ) : Word
10496: Function GetWindowWord( hWnd : HWND; nIndex : Integer ) : Word
10497: Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word ) : Word
10498: Function GetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
10499: Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint

```

```

10500: Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10501: Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10502: Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10503: Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10504: Function GetDesktopWindow : HWND
10505: Function GetParent( hWnd : HWND) : HWND
10506: Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10507: Function GetTopWindow( hWnd : HWND) : HWND
10508: Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10509: Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10510: //Delphi DFM
10511: Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10512: Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10513: procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10514: function GetHighlightersFilter(AHighlighters: TStringList): string;
10515: function GetHighlighterFromExt(AHighlighters: TStringList; Extension: string):TSynCustomHighlighter;
10516: Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10517: Function OpenIcon( hWnd : HWND) : BOOL
10518: Function CloseWindow( hWnd : HWND) : BOOL
10519: Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10520: Function SetWindowPos(hWnd:HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UInt) : BOOL
10521: Function IsWindowVisible( hWnd : HWND) : BOOL
10522: Function IsIconic( hWnd : HWND) : BOOL
10523: Function AnyPopup : BOOL
10524: Function BringWindowToTop( hWnd : HWND) : BOOL
10525: Function IsZoomed( hWnd : HWND) : BOOL
10526: Function IsWindow( hWnd : HWND) : BOOL
10527: Function IsMenu( hMenu : HMENU) : BOOL
10528: Function IsChild( hWndParent, hWnd : HWND) : BOOL
10529: Function DestroyWindow( hWnd : HWND) : BOOL
10530: Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10531: Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10532: Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10533: Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10534: Function IsWindowUnicode( hWnd : HWND) : BOOL
10535: Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10536: Function IsWindowEnabled( hWnd : HWND) : BOOL
10537:
10538: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10539: begin
10540:   const ('ShowSetupDialogOptLong','String '--setup
10541: PrimaryConfPathOptLong','String '--primary-config-path=
10542: PrimaryConfPathOptShort','String '--pcp=
10543: SecondaryConfPathOptLong','String '--secondary-config-path=
10544: SecondaryConfPathOptShort','String '--scp=
10545: NoSplashScreenOptLong','String '--no-splash-screen
10546: NoSplashScreenOptShort','String '--nsc
10547: StartedByStartLazarusOpt','String '--started-by-startlazarus
10548: SkipLastProjectOpt','String '--skip-last-project
10549: DebugLogOpt','String '--debug-log=
10550: DebugLogOptEnable','String '--debug-enable=
10551: LanguageOpt','String '--language=
10552: LazarusDirOpt','String '--lazarusdir=
10553: Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10554: Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10555: Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10556: Function IsHelpRequested : Boolean
10557: Function IsVersionRequested : boolean
10558: Function GetLanguageSpecified : string
10559: Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10560: Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10561: Procedure ParseNoGuiCmdlineParams
10562: Function ExtractCmdLineFilenames : TStrings
10563: end;
10564:
10565:
10566: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10567: begin
10568:   Function CompareFilenames( const Filenam1, Filenam2 : string) : integer
10569:   Function CompareFilenamesIgnoreCase( const Filenam1, Filenam2 : string) : integer
10570:   Function CompareFileExt( const Filenam, Ext : string; CaseSensitive : boolean) : integer;
10571:   Function CompareFileExt1( const Filenam, Ext : string) : integer;
10572:   Function CompareFilenameStarts( const Filenam1, Filenam2 : string) : integer
10573:   Function CompareFilenames(Filenam1:PChar;Len1:integer; Filenam2:PChar;Len2:integer):integer
10574:   Function CompareFilenamesP( Filenam1, Filenam2 : PChar; IgnoreCase : boolean) : integer
10575:   Function DirPathExists( DirectoryName : string) : boolean
10576:   Function DirectoryIsWritable( const DirectoryName : string) : boolean
10577:   Function ExtractFileNameOnly( const AFilename : string) : string
10578:   Function FilenameIsAbsolute( const Thefilename : string) : boolean
10579:   Function FilenameIsWinAbsolute( const Thefilename : string) : boolean
10580:   Function FilenameIsUnixAbsolute( const Thefilename : string) : boolean
10581:   Function ForceDirectory(DirectoryName : string) : boolean
10582:   Procedure CheckIfFileIsExecutable( const AFilename : string)
10583:   Procedure CheckIfFileIsSymlink( const AFilename : string)
10584:   Function FileIsText( const AFilename : string) : boolean
10585:   Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10586:   Function FilenameIsTrimmed( const Thefilename : string) : boolean
10587:   Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10588:   Function TrimFilename( const AFilename : string) : string

```

```

10589: Function ResolveDots( const AFilename : string ) : string
10590: Procedure ForcePathDelims( var FileName : string )
10591: Function GetForcedPathDelims( const FileName : string ) : String
10592: Function CleanAndExpandfilename( const Filename : string ) : string
10593: Function CleanAndExpandDirectory( const Filename : string ) : string
10594: Function TrimAndExpandfilename( const Filename : string; const BaseDir : string ) : string
10595: Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string ) : string
10596: Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10597: Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
AlwaysRequireSharedBaseFolder: Boolean) : string
10598: Function FileIsInPath( const Filename, Path : string ) : boolean
10599: Function AppendPathDelim( const Path : string ) : string
10600: Function ChompPathDelim( const Path : string ) : string
10601: Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string ) : string
10602: Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string ) : string
10603: Function MinimizeSearchPath( const SearchPath : string ) : string
10604: Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
(*Function FileExistsUTF8( const Filename : string ) : boolean
10605: Function FileAgeUTF8( const FileName : string ) : Longint
10606: Function DirectoryExistsUTF8( const Directory : string ) : Boolean
10607: Function ExpandFileNameUTF8( const FileName : string; BaseDir : string ) : string
10608: Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10609: Function FindNextUTF8( var Rslt : TSearchRec ) : Longint
10610: Function FindCloseUTF8( var F : TSearchrec )
10611: Procedure FindCloseUTF8( var F : TSearchrec )
10612: Function FileSetDateUTF8( const FileName : String; Age : Longint ) : Longint
10613: Function FileGetAttrUTF8( const FileName : String ) : Longint
10614: Function FileSetAttrUTF8( const Filename : String; Attr : longint ) : Longint
10615: Function DeleteFileUTF8( const FileName : String ) : Boolean
10616: Function RenameFileUTF8( const OldName, NewName : String ) : Boolean
10617: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean ) : String
10618: Function FileIsReadOnlyUTF8( const FileName : String ) : Boolean
10619: Function GetCurrentDirUTF8 : String
10620: Function SetCurrentDirUTF8( const NewDir : String ) : Boolean
10621: Function CreateDirUTF8( const NewDir : String ) : Boolean
10622: Function RemoveDirUTF8( const Dir : String ) : Boolean
10623: Function ForceDirectoriesUTF8( const Dir : string ) : Boolean
10624: Function FileOpenUTF8( const FileName : string; Mode : Integer ) : THandle
10625: Function FileCreateUTF8( const FileName : string ) : THandle;
10626: Function FileCreateUTF81( const FileName : string; Rights : Cardinal ) : THandle;
10627: Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal ) : THandle;
10628: Function FileSizeUtf8( const Filename : string ) : int64
10629: Function GetfileDescription( const AFilename : string ) : string
10630: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean ) : string
10631: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean ) : string
10632: Function GetTempFileNameUTF8( const Dir, Prefix : String ) : String*)
10633: Function IsUNCPath( const Path : String ) : Boolean
10634: Function ExtractUNCVolume( const Path : String ) : String
10635: Function ExtractFileRoot( FileName : String ) : String
10636: Function GetDarwinSystemFilename( Filename : string ) : string
10637: Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean )
10638: Function StrToCmdLineParam( const Param : string ) : string
10639: Function MergeCmdLineParams( ParamList : TStrings ) : string
10640: Procedure InvalidateFileStateCache( const Filename : string )
10641: Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList;
10642: Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean ) : TStringList
10643: Function FindAllDocs(const Root, extmask: string): TStringlist;
10644: Function ReadfileToString( const Filename : string ) : string
10645: procedure Incl(var X: longint; N: Longint);
10646:
10647: type
10648: TCopyFileFlag = ( cffOverwriteFile,
10649:                   cffCreateDestDirectory, cffPreserveTime );
10650: TCopyFileFlags = set of TCopyFileFlag;*
10651: TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10652: TCopyFileFlags', 'set of TCopyFileFlag
10653: Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags ) : Boolean
10654: end;
10655:
10656: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10657: begin
10658:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10659:   SIRegister_TMask(CL);
10660:   SIRegister_TParseStringList(CL);
10661:   SIRegister_TMaskList(CL);
10662:   Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean ) : Boolean
10663:   Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean ) : Bool;
10664:   Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10665:   Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10666: end;
10667:
10668: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10669: begin
10670:   //PShellHookInfo', '^TShellHookInfo // will not work
10671:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10672:   SHELLHOOKINFO', 'TShellHookInfo
10673:   LPSHELLHOOKINFO', 'PShellHookInfo
10674:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10675:   SIRegister_TJvShellHook(CL);

```

```

10676: Function InitJvShellHooks : Boolean
10677: Procedure UnInitJvShellHooks
10678: end;
10679:
10680: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10681: begin
10682:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10683:   +', dcHasSelSel, dcWantTab, dcNative )
10684:   TDlgCodes', 'set of TDlgCode
10685: 'dcWantMessage', ' dcWantAllKeys);
10686: SIRegister_IJVExControl(CL);
10687: SIRegister_IJVDenySubClassing(CL);
10688: SIRegister_TStructPtrMessage(CL);
10689: Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10690: Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10691: Procedure DrawDotNetControl( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10692: Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10693: Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint ) : TMessage;
10694: Function CreateWMMessagel( Msg : Integer; WParam : Integer; LParam : TControl ) : TMessage;
10695: Function SmallPointToLong( const Pt : TSmallPoint ) : Longint
10696: Function ShiftStateToKeyData( Shift : TShiftState ) : Longint
10697: Function GetFocusedControl( AControl : TControl ) : TWinControl
10698: Function DlgcToDlgCodes( Value : Longint ) : TDlgCodes
10699: Function DlgCodesToDlgc( Value : TDlgCodes ) : Longint
10700: Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10701: Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage ) : Boolean
10702: SIRegister_TJVExControl(CL);
10703: SIRegister_TJVExWinControl(CL);
10704: SIRegister_TJVExCustomControl(CL);
10705: SIRegister_TJVExGraphicControl(CL);
10706: SIRegister_TJVExHintWindow(CL);
10707: SIRegister_TJVExPubGraphicControl(CL);
10708: end;
10709:
10710: (*-----*)
10711: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10712: begin
10713:   Procedure EncodeStream( Input, Output : TStream)
10714:   Procedure DecodeStream( Input, Output : TStream)
10715:   Function EncodeStringl( const Input : string) : string
10716:   Function DecodeStringl( const Input : string) : string
10717: end;
10718:
10719: (*-----*)
10720: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10721: begin
10722:   SIRegister_TWebAppRegInfo(CL);
10723:   SIRegister_TWebAppRegList(CL);
10724:   Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10725:   Procedure RegisterWebApp( const AFileName, AProgID : string)
10726:   Procedure UnregisterWebApp( const AProgID : string)
10727:   Function FindRegisteredWebApp( const AProgID : string) : string
10728:   Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10729:   'sUDPPort','String 'UDPPort
10730: end;
10731:
10732: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10733: begin
10734:   // TStringDynArray', 'array of string
10735:   Function GetEnvVarValue( const VarName : string) : string
10736:   Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10737:   Function DeleteEnvVar( const VarName : string) : Integer
10738:   Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
BufSize:Int):Int;
10739:   Function ExpandEnvVars( const Str : string) : string
10740:   Function GetAllEnvVars( const Vars : TStrings) : Integer
10741:   Procedure GetAllEnvVarNames( const Names : TStrings);
10742:   Function GetAllEnvVarNames1 : TStringDynArray;
10743:   Function EnvBlockSize : Integer
10744:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10745:   SIRegister_TPJEnvVarsEnumerator(CL);
10746:   SIRegister_TPJEnvVars(CL);
10747:   FindClass('TOBJECT'),'EPJEnvVars
10748:   FindClass('TOBJECT'),'EPJEnvVars
10749:   //Procedure Register
10750: end;
10751:
10752: (*-----*)
10753: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10754: begin
10755:   'cOneSecInMS','LongInt'( 1000 );
10756:   // 'cDefTimeSlice','LongInt'( 50 );
10757:   // 'cDefMaxExecTime',' cOneMinInMS';
10758:   'cAppErrorMask','LongInt'( 1 shl 29 );
10759:   Function IsApplicationError( const ErrCode : LongWord ) : Boolean
10760:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10761:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10762:   Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10763:   Function MakeConsoleColors1( const AForeground, ABackground : TColor ) : TPJConsoleColors;

```

```

10764: Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor ) : TPJConsoleColors;
10765: Function MakeSize( const ACX, ACY : LongInt ) : TSize
10766: SIRegister_TPJCustomConsoleApp(CL);
10767: SIRegister_TPJConsoleApp(CL);
10768: end;
10769:
10770: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10771: begin
10772:   INVALID_IP_ADDRESS', 'LongWord').SetUInt( $ffffffff );
10773:   t_encoding', '( uuencode, base64, mime )
10774:   Function internet_date( date : TDateTime ) : string
10775:   Function lookup_hostname( const hostname : string ) : longint
10776:   Function my_hostname : string
10777:   Function my_ip_address : longint
10778:   Function ip2string( ip_address : longint ) : string
10779:   Function resolve_hostname( ip : longint ) : string
10780:   Function address_from( const s : string; count : integer ) : string
10781:   Function encode_base64( data : TStream ) : TStringList
10782:   Function decode_base64( source : TStringList ) : TMemoryStream
10783:   Function posn( const s, t : string; count : integer ) : integer
10784:   Function poscn( c : char; const s : string; n : integer ) : integer
10785:   Function filename_of( const s : string ) : string
10786: //Function trim( const s : string ) : string
10787: //Procedure setlength( var s : string; l : byte )
10788:   Function TimeZoneBias : longint
10789:   Function eight2seven_quoteprint( const s : string ) : string
10790:   Function eight2seven_german( const s : string ) : string
10791:   Function seven2eight_quoteprint( const s : string ) : string end;
10792:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10793:   Function socketerror : cint
10794:   Function fpsocket( domain : cint; xtype : cint; protocol : cint ) : cint
10795:   Function frecv( s : cint; buf : __pointer; len : size_t; flags : cint ) : ssize_t
10796:   Function fpsend( s : cint; msg : __pointer; len : size_t; flags : cint ) : ssize_t
10797: //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen ) : cint
10798:   Function fplisten( s : cint; backlog : cint ) : cint
10799: //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint ) : cint
10800: //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen ) : cint
10801: //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen ) : cint
10802:   Function NetAddrToStr( Entry : in_addr ) : String
10803:   Function HostAddrToStr( Entry : in_addr ) : String
10804:   Function StrToHostAddr( IP : String ) : in_addr
10805:   Function StrToNetAddr( IP : String ) : in_addr
10806:   SOL_SOCKET', 'LongWord').SetUInt( $ffff );
10807:   cint8', 'shortint
10808:   cuint8', 'byte
10809:   cchar', 'cint8
10810:   cschar', 'cint8
10811:   cuchar', 'cuint8
10812:   cint16', 'smallint
10813:   cuint16', 'word
10814:   cshort', 'cint16
10815:   cshort', 'cint16
10816:   c ushort', 'cuint16
10817:   cint32', 'longint
10818:   cuint32', 'longword
10819:   cint', 'cint32
10820:   csint', 'cint32
10821:   cuint', 'cuint32
10822:   csigned', 'cint
10823:   cunsigned', 'cuint
10824:   cint64', 'int64
10825:   clonglong', 'cint64
10826:   cslonglong', 'cint64
10827:   cbool', 'longbool
10828:   cfloat', 'single
10829:   cdouble', 'double
10830:   clongdouble', 'extended
10831:
10832: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10833: begin
10834:   TlkJSONTypes', '( jsBase, jsNumber, jsString, jsBoolean, jsNull, jsList, jsObject )
10835:   SIRegister_TlkJSONdotnetclass(CL);
10836:   SIRegister_TlkJSONbase(CL);
10837:   SIRegister_TlkJSONnumber(CL);
10838:   SIRegister_TlkJSONstring(CL);
10839:   SIRegister_TlkJSONboolean(CL);
10840:   SIRegister_TlkJSONnull(CL);
10841:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Ele : TlkJSONba'
10842:     + 'se; data : TObject; var Continue : Boolean)
10843:   SIRegister_TlkJSONcustomlist(CL);
10844:   SIRegister_TlkJSONlist(CL);
10845:   SIRegister_TlkJSONobjectmethod(CL);
10846:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10847:   TlkHashFunction', 'Function ( const ws : WideString ) : cardinal
10848:   SIRegister_TlkHashTable(CL);
10849:   SIRegister_TlkBalTree(CL);
10850:   SIRegister_TlkJSONObject(CL);
10851:   SIRegister_TlkJSON(CL);
10852:   SIRegister_TlkJSONstreamed(CL);

```

```

10853: Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10854: end;
10855:
10856: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10857: begin
10858:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10859:   SIRegister_TZSortedList(CL);
10860:   Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10861:   Function zLastDelimiter( const Delimiters, Str : string) : Integer
10862:   //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10863:   //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10864:   Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10865:   Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10866:   Function EndsWith( const Str, SubStr : WideString) : Boolean;
10867:   Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10868:   Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10869:   Function SQLStrToFloatDef1( Str : string; Def : Extended) : Extended;
10870:   Function SQLStrToFloat( const Str : AnsiString) : Extended
10871:   //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10872:   //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10873:   Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10874:   Function StrToBoolEx( Str : string) : Boolean
10875:   Function BoolToStrEx( Bool : Boolean) : String
10876:   Function IsIpAddr( const Str : string) : Boolean //IsIP()
10877:   Function zSplitString( const Str, Delimiters : string) : TStrings
10878:   Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10879:   Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10880:   Function ComposeString( List : TStrings; const Delimiter : string) : string
10881:   Function FloatToSQLStr( Value : Extended) : string
10882:   Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10883:   Function SplitStringEx( const Str, Delimiter : string) : TStrings
10884:   Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10885:   Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10886:   Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10887:   Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10888:   Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10889:   Function StrToBytes3( const Value : WideString) : TByteDynArray;
10890:   Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10891:   Function BytesToVar( const Value : TByteDynArray) : Variant
10892:   Function VarToBytes( const Value : Variant) : TByteDynArray
10893:   Function AnsiSQLDateToDate( const Value : string) : TDateTime
10894:   Function TimestampStrToDate( const Value : string) : TDateTime
10895:   Function DateToString( Value : TDateTime; WithMMSec : Boolean) : string
10896:   Function EncodeCString( const Value : string) : string
10897:   Function DecodeCString( const Value : string) : string
10898:   Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10899:   Function MemPas( Buffer : PChar; Length : LongInt) : string
10900:   Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
SubVersion:Int);
10901:   Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
SubVersion:Integer);int;
10902:   Function FormatsSQLVersion( const SQLVersion : Integer ) : String
10903:   Function ZStrToFloat( Value : AnsiChar ) : Extended;
10904:   Function ZStrToFloat1( Value : AnsiString ) : Extended;
10905:   Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString );
10906:   Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString );
10907:   Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String );
10908:   Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String );
10909:   Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString );
10910:   Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString );
10911:   Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString );
10912: end;
10913:
10914: unit uPSI_ZEncoding;
10915: Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word ) : RawByteString
10916: Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word ) : String
10917: Function ZRawToUnicode( const S : RawByteString; const CP : Word ) : WideString
10918: Function ZUnicodeToRaw( const US : WideString; CP : Word ) : RawByteString
10919: Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word ) : RawByteString
10920: Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word ) : AnsiString
10921: Function ZConvertAnsiToUTF8( const Src : AnsiString ) : UTF8String
10922: Function ZConvertUTF8ToAnsi( const Src : UTF8String ) : AnsiString
10923: Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word ) : UTF8String
10924: Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word ) : RawByteString
10925: Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word ) : String
10926: Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word ) : RawByteString
10927: Function ZConvertStringToRawWithAutoEncode( const Src:String;const StringCP,RawCP:Word):RawByteString;
10928: Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word ) : String
10929: Function ZConvertStringToUTF8( const Src : String; const StringCP : Word ) : UTF8String
10930: Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP : Word): UTF8String
10931: Function ZConvertStringToAnsi( const Src : String; const StringCP : Word ) : AnsiString
10932: Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP : Word): AnsiString
10933: Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word ) : String
10934: Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word ) : String
10935: Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word ) : String
10936: Function ZConvertStringToUnicode( const Src : String; const StringCP : Word ) : WideString
10937: Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word ) : WideString
10938: Function ZConvertStringToUnicodeWithAutoEncode( const Src : String; const StringCP:Word):WideString
10939: Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word ) : RawByteString

```

```

10940: Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10941: Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10942: Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10943: Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10944: Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10945: Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10946: Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10947: Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10948: Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10949: Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10950: Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10951: Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10952: Function ZUnknownRawToUnicodeWithAutoEncode( 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;
10957:
10958: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10959: begin
10960:   ('RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10961:   ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10962:   ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10963:   ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10964:   ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10965:   ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10966:   ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10967:   {'alDefault', '1 RPC_C_AUTHN_LEVEL_DEFAULT};
10968:   {'alNone', '2 RPC_C_AUTHN_LEVEL_NONE};
10969:   {'alConnect', '3 RPC_C_AUTHN_LEVEL_CONNECT};
10970:   {'alCall', '4 RPC_C_AUTHN_LEVEL_CALL};
10971:   {'alPacket', '5 RPC_C_AUTHN_LEVEL_PKT};
10972:   {'alPacketIntegrity', '6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY};
10973:   {'alPacketPrivacy', '7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY};
10974:   ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10975:   ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10976:   ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10977:   ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10978:   ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10979:   {'ilDefault', '0 RPC_C_IMP_LEVEL_DEFAULT};
10980:   {'ilAnonymous', '1 RPC_C_IMP_LEVEL_ANONYMOUS};
10981:   {'ilIdentiry', '2 RPC_C_IMP_LEVEL_IDENTIFY};
10982:   {'ilImpersonate', '3 RPC_C_IMP_LEVEL_IMPERSONATE};
10983:   {'ilDelegate', '4 RPC_C_IMP_LEVEL_DELEGATE};
10984:   ('EOAC_NONE','LongWord').SetUInt( $0);
10985:   ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10986:   ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10987:   ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10988:   ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10989:   ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10990:   ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10991:   ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10992:   ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10993:   ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10994:   FindClass('TOBJECT'),'EBoldCom
10995: Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10996: Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10997: Function BoldStreamToVariant( Stream : TStream) : OleVariant
10998: Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10999: Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
11000: Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
11001: Function BoldVariantArrayOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
11002: Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
11003: Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
11004: Function BoldGetValue( Data : OleVariant; const Name : string) : OleVariant
11005: Procedure BoldSetValue( Data : OleVariant; const Name : string; Value : OleVariant)
11006: Function BoldCreateGUID : TGUID
11007: Function BoldCreateComObject( const ClsId, IID : TGUID; out Obj : variant; out Res : HResult) : Boolean
11008: Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IID:GUID;out Obj:variant;out Res:HRes):Bool;
11009: Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
11010: Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
11011: end;
11012:
11013: (*-----*)
11014: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11015: begin
11016:   Function ParseISODate( s : string) : TDateTime
11017:   Function ParseISODateTime( s : string) : TDateTime
11018:   Function ParseISOTime( str : string) : TDateTime
11019:   end;
11020:
11021: (*-----*)
11022: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11023: begin
11024:   Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
11025:   Function BoldCreateGUIDWithBracketsAsString : string
11026:   end;
11027:

```

```

11028: procedure SIRегистер_BoldFileHandler(CL: TPSPPascalCompiler);
11029: begin
11030:   FindClass('TOBJECT'), 'TBoldFileHandler'
11031:   FindClass('TOBJECT'), 'TBoldDiskFileHandler'
11032:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11033:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList )
11034:   SIRегистер_TBoldFileHandler(CL);
11035:   SIRегистер_TBoldDiskFileHandler(CL);
11036:   Procedure BoldCloseAllFilehandlers
11037:   Procedure BoldRemoveUnchangedFilesFromEditor
11038:   Function BoldFileHandlerList : TBoldObjectArray
11039:   Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
11040:   OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
11041: end;
11042: procedure SIRегистер_BoldWinINet(CL: TPSPPascalCompiler);
11043: begin
11044:   PCharArr', 'array of PChar
11045:   Function BoldInternetOpen(Agent:String;
11046:   AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr;
11047:   Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11048:   NumberofBytesRead:Card):LongBool;
11049:   Function BoldInternetCloseHandle( HINet : Pointer ) : LongBool
11050:   Cardinal; Reserved : Cardinal) : LongBool
11051:   Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
11052:   Cardinal; Context : Cardinal ) : LongBool
11053:   Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
11054:   : PCharArr; Flags, Context : Cardinal) : Pointer
11055:   Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
11056:   Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11057:   Function BoldInternetAttemptConnect( dwReserved : DWORD ) : DWORD
11058:   Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
11059:   Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11060:   Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11061: end;
11062: procedure SIRегистер_BoldQueryUserDlg(CL: TPSPPascalCompiler);
11063: begin
11064:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11065:   SIRегистер_TfrmBoldQueryUser(CL);
11066:   Function QueryUser( const Title, Query : string ) : TBoldQueryResult
11067: (*-----*)
11068: procedure SIRегистер_BoldQueue(CL: TPSPPascalCompiler);
11069: begin
11070:   //('befIsInDisplayList',' BoldElementFlag0 );
11071:   //('befStronglyDependedOfPrioritized',' BoldElementFlag1 );
11072:   //('befFollowerSelected',' BoldElementFlag2 );
11073:   FindClass('TOBJECT'), 'TBoldQueue
11074:   FindClass('TOBJECT'), 'TBoldQueueable
11075:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11076:   SIRегистер_TBoldQueueable(CL);
11077:   Function BoldQueueFinalized : Boolean
11078:   Function BoldInstalledQueue : TBoldQueue
11079: end;
11080:
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:
11102: procedure SIRегистер_Geometry(CL: TPSPPascalCompiler); //OpenGL
11103: begin
11104:   THomogeneousByteVector', 'array[0..3] of Byte
11105:   THomogeneousWordVector', 'array[0..3] of Word
11106:   THomogeneousIntVector', 'array[0..3] of Integer
11107:   THomogeneousFltVector', 'array[0..3] of single
11108:   THomogeneousDblVector', 'array[0..3] of double
11109:   THomogeneousExtVector', 'array[0..3] of extended

```

```

11110: TAffineByteVector', 'array[0..2] of Byte
11111: TAffineWordVector', 'array[0..2] of Word
11112: TAffineIntVector', 'array[0..2] of Integer
11113: TAffineFltVector', 'array[0..2] of single
11114: TAffineDblVector', 'array[0..2] of double
11115: TAffineExtVector', 'array[0..2] of extended
11116: THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11117: THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11118: THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11119: THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11120: THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11121: THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11122: TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11123: TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11124: TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11125: TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11126: TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11127: TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11128: TMatrix4b', 'THomogeneousByteMatrix
11129: TMatrix4w', 'THomogeneousWordMatrix
11130: TMatrix4i', 'THomogeneousIntMatrix
11131: TMatrix4f', 'THomogeneousFltMatrix
11132: TMatrix4d', 'THomogeneousDblMatrix
11133: TMatrix4e', 'THomogeneousExtMatrix
11134: TMatrix3b', 'TAffineByteMatrix
11135: TMatrix3w', 'TAffineWordMatrix
11136: TMatrix3i', 'TAffineIntMatrix
11137: TMatrix3f', 'TAffineFltMatrix
11138: TMatrix3d', 'TAffineDblMatrix
11139: TMatrix3e', 'TAffineExtMatrix
11140: //PMatrix', '^TMatrix // will not work
11141: TMatrixGL', 'THomogeneousFltMatrix
11142: THomogeneousMatrix', 'THomogeneousFltMatrix
11143: TAffineMatrix', 'TAffineFltMatrix
11144: TQuaternion', 'record Vector : TVector4f; end
11145: TRectangle', 'record Left : integer; Top : integer; Width : integer;
11146: +ger; Height : Integer; end
11147: TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11148: +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11149: +' , ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11150: 'EPSILON', 'Extended').setExtended( 1E-100 );
11151: 'EPSILON2', 'Extended').setExtended( 1E-50 );
11152: Function VectorAddGL( V1, V2 : TVectorGL ) : TVectorGL
11153: Function VectorAffineAdd( V1, V2 : TAffineVector ) : TAffineVector
11154: Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11155: Function VectorAffineDotProduct( V1, V2 : TAffineVector ) : Single
11156: Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single ) : TAffineVector
11157: Function VectorAffineSubtract( V1, V2 : TAffineVector ) : TAffineVector
11158: Function VectorAngle( V1, V2 : TAffineVector ) : Single
11159: Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single ) : TVectorGL
11160: Function VectorCrossProduct( V1, V2 : TAffineVector ) : TAffineVector
11161: Function VectorDotProduct( V1, V2 : TVectorGL ) : Single
11162: Function VectorLength( V : array of Single ) : Single
11163: Function VectorLerp( V1, V2 : TVectorGL; t : Single ) : TVectorGL
11164: Procedure VectorNegate( V : array of Single )
11165: Function VectorNorm( V : array of Single ) : Single
11166: Function VectorNormalize( V : array of Single ) : Single
11167: Function VectorPerpendicular( V, N : TAffineVector ) : TAffineVector
11168: Function VectorReflect( V, N : TAffineVector ) : TAffineVector
11169: Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single )
11170: Procedure VectorScale( V : array of Single; Factor : Single )
11171: Function VectorSubtractGL( V1, V2 : TVectorGL ) : TVectorGL
11172: Function CreateRotationMatrixX( Sine, Cosine : Single ) : TMatrixGL
11173: Function CreateRotationMatrixY( Sine, Cosine : Single ) : TMatrixGL
11174: Function CreateRotationMatrixZ( Sine, Cosine : Single ) : TMatrixGL
11175: Function CreateScaleMatrix( V : TAffineVector ) : TMatrixGL
11176: Function CreateTranslationMatrix( V : TVectorGL ) : TMatrixGL
11177: Procedure MatrixAdjoint( var M : TMatrixGL )
11178: Function MatrixAffineDeterminant( M : TAffineMatrix ) : Single
11179: Procedure MatrixAffineTranspose( var M : TAffineMatrix )
11180: Function MatrixDeterminant( M : TMatrixGL ) : Single
11181: Procedure MatrixInvert( var M : TMatrixGL )
11182: Function MatrixMultiply( M1, M2 : TMatrixGL ) : TMatrixGL
11183: Procedure MatrixScale( var M : TMatrixGL; Factor : Single )
11184: Procedure MatrixTranspose( var M : TMatrixGL )
11185: Function QuaternionConjugate( Q : TQuaternion ) : TQuaternion
11186: Function QuaternionFromPoints( V1, V2 : TAffineVector ) : TQuaternion
11187: Function QuaternionMultiply( qL, qR : TQuaternion ) : TQuaternion
11188: Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11189: Function QuaternionToMatrix( Q : TQuaternion ) : TMatrixGL
11190: Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector )
11191: Function ConvertRotation( Angles : TAffineVector ) : TVectorGL
11192: Function CreateRotationMatrix( Axis : TVector3f; Angle : Single ) : TMatrixGL
11193: //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations ) : Boolean
11194: Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix ) : TAffineVector
11195: Function VectorTransform( V : TVector4f; M : TMatrixGL ) : TVector4f;
11196: Function VectorTransform1( V : TVector3f; M : TMatrixGL ) : TVector3f;
11197: Function MakeAffineDblVector( V : array of Double ) : TAffineDblVector
11198: Function MakeDblVector( V : array of Double ) : THomogeneousDblVector

```

```

11199: Function MakeAffineVector( V : array of Single ) : TAffineVector
11200: Function MakeQuaternion( Imag : array of Single; Real : Single ) : TQuaternion
11201: Function MakeVector( V : array of Single ) : TVectorGL
11202: Function PointInPolygonGL( xp, yp : array of Single; x, y : Single ) : Boolean
11203: Function VectorAffineDblToFlt( V : TAffineDblVector ) : TAffineVector
11204: Function VectorDblToFlt( V : THomogeneousDblVector ) : THomogeneousVector
11205: Function VectorAffineFltToDbl( V : TAffineVector ) : TAffineDblVector
11206: Function VectorFltToDbl( V : TVectorGL ) : THomogeneousDblVector
11207: Function ArcCosGL( X : Extended ) : Extended
11208: Function ArcSingL( X : Extended ) : Extended
11209: Function ArcTan2GL( Y, X : Extended ) : Extended
11210: Function CoTanGL( X : Extended ) : Extended
11211: Function DegToRadGL( Degrees : Extended ) : Extended
11212: Function RadToDegGL( Radians : Extended ) : Extended
11213: Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended )
11214: Function TanGL( X : Extended ) : Extended
11215: Function Turn( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11216: Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11217: Function Pitch( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11218: Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11219: Function Roll( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11220: Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11221: end;
11222:
11223:
11224: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11225: begin
11226:   Function RegCreateKey( const RootKey : HKEY; const Key, Value : string ) : Longint
11227:   Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string ) : Boolean
11228:   Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string ) : Boolean
11229:   Function RegReadBool( const RootKey : HKEY; const Key, Name : string ) : Boolean
11230:   Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean ) : Boolean
11231:   Function RegReadInteger( const RootKey : HKEY; const Key, Name : string ) : Integer
11232:   Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer ) : Integer
11233:   Function RegReadString( const RootKey : HKEY; const Key, Name : string ) : string
11234:   Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string ) : string
11235:   Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string ) : Int64
11236:   Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64 ) : Int64
11237:   Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean )
11238:   Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer )
11239:   Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string )
11240:   Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64 )
11241:   Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings ) : Boolean
11242:   Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings ) : Boolean
11243:   Function RegHasSubKeys( const RootKey : HKEY; const Key : string ) : Boolean
11244:   Function RegKeyExists( const RootKey : HKEY; const Key : string ) : Boolean
11245:   AddTypeS('TExecKind', ['ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11246:     +'RunOnce, ekServiceRun, ekServiceRunOnce ')
11247:   AddClassN(FindClass('TOBJECT'), EJclRegistryError
11248:   Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string ) : Boolean
11249:   Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string ) : Boolean
11250:   Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
11251: Items:TStrings):Bool;
11251:   Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
11251: SaveTo:TStrings):Bool;
11252:   Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11253: end;
11254:
11255: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11256: begin
11257:   CLSID_StdComponentCategoriesMgr', 'GUID '{0002E005-0000-0000-C000-00000000046}
11258:   CATID_SafeForInitializing', 'GUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11259:   CATID_SafeForScripting', 'GUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11260:   icMAX_CATEGORY_DESC_LEN', 'LongInt'( 128 );
11261:   FindClass('TOBJECT'), 'EInvalidParam
11262:   Function IsDCOMInstalled : Boolean
11263:   Function IsDCOMEnabled : Boolean
11264:   Function GetDCOMVersion : string
11265:   Function GetMDACVersion : string
11266:   Function GetMDACVersion2 : string
11267:   Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
11267: VarArray:OleVariant):HRESULT;
11268:   Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HRESULT;
11269:   Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
11269: VarArray:OleVariant):HRESULT;
11270:   Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HRESULT;
11271:   Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
11271: VarArray:OleVariant):HRESULT;
11272:   Function CreateComponentCategory( const CatID : TGUID; const sDescription : string ) : HRESULT
11273:   Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HRESULT
11274:   Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HRESULT
11275:   Function ResetIStreamToStart( Stream : IStream ) : Boolean
11276:   Function SizeOfIStreamContents( Stream : IStream ) : Largeint
11277:   Function StreamToVariantArray( Stream : TStream ) : OleVariant;
11278:   Function StreamToVariantArray1( Stream : IStream ) : OleVariant;
11279:   Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream );
11280:   Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream );
11281: end;
11282:

```

```

11283:
11284: procedure SIRegister_JclUnitConv_mX2(CL: TPSCompiler);
11285: begin
11286:   Const ('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11287:   FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11288:   KelvinFreezingPoint','Extended').setExtended( 273.15);
11289:   CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11290:   FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11291:   KelvinAbsoluteZero','Extended').setExtended( 0.0);
11292:   DegPerCycle','Extended').setExtended( 360.0);
11293:   DegPerGrad','Extended').setExtended( 0.9);
11294:   DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11295:   GradPerCycle','Extended').setExtended( 400.0);
11296:   GradPerDeg','Extended').setExtended( 1.11111111111111111111111111111111);
11297:   GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11298:   RadPerCycle ','Extended').setExtended( 6.283185307179586476925286766559);
11299:   RadPerDeg ','Extended').setExtended( 0.017453292519943295769236907684886);
11300:   RadPerGrad ','Extended').setExtended( 0.015707963267948966192313216916398);
11301:   CyclePerDeg ','Extended').setExtended( 0.0027777777777777777777777777777778);
11302:   CyclePerGrad ','Extended').setExtended( 0.0025);
11303:   CyclePerRad ','Extended').setExtended( 0.15915494309189533576888376337251);
11304:   ArcMinutesPerDeg ','Extended').setExtended( 60.0);
11305:   ArcSecondsPerArcMinute ','Extended').setExtended( 60.0);
11306:   Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11307:   Function MakePercentage( const Step, Max : Longint) : Longint
11308:   Function CelsiusToKelvin( const T : double) : double
11309:   Function CelsiusToFahrenheit( const T : double) : double
11310:   Function KelvinToCelsius( const T : double) : double
11311:   Function KelvinToFahrenheit( const T : double) : double
11312:   Function FahrenheitToCelsius( const T : double) : double
11313:   Function FahrenheitToKelvin( const T : double) : double
11314:   Function CycleToDeg( const Cycles : double) : double
11315:   Function CycleToGrad( const Cycles : double) : double
11316:   Function CycleToRad( const Cycles : double) : double
11317:   Function DegToCycle( const Degrees : double) : double
11318:   Function DegToGrad( const Degrees : double) : double
11319:   Function DegToRad( const Degrees : double) : double
11320:   Function GradToCycle( const Grads : double) : double
11321:   Function GradToDeg( const Grads : double) : double
11322:   Function GradToRad( const Grads : double) : double
11323:   Function RadToCycle( const Radians : double) : double
11324:   Function RadToDeg( const Radians : double) : double
11325:   Function RadToGrad( const Radians : double) : double
11326:   Function DmsToDeg( const D, M : Integer; const S : double) : double
11327:   Function DmsToRad( const D, M : Integer; const S : double) : double
11328:   Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11329:   Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11330:   Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11331:   Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11332:   Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11333:   Procedure CartesianToSpherical( const X, Y, Z : double; out Rho, Phi, Theta : double)
11334:   Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11335:   Procedure SphericalToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11336:   Function CmToInch( const Cm : double) : double
11337:   Function InchToCm( const Inch : double) : double
11338:   Function FeetToMetre( const Feet : double) : double
11339:   Function MetreToFeet( const Metre : double) : double
11340:   Function YardToMetre( const Yard : double) : double
11341:   Function MetreToYard( const Metre : double) : double
11342:   Function NmToKm( const Nm : double) : double
11343:   Function KmToNm( const Km : double) : double
11344:   Function KmToSm( const Km : double) : double
11345:   Function SmToKm( const Sm : double) : double
11346:   Function LitreToGalUs( const Litre : double) : double
11347:   Function GalUsToLitre( const GalUs : double) : double
11348:   Function GalUsToGalCan( const GalUs : double) : double
11349:   Function GalCanToGalUs( const GalCan : double) : double
11350:   Function GalUsToGalUk( const GalUs : double) : double
11351:   Function GalUkToGalUs( const GalUk : double) : double
11352:   Function LitreToGalCan( const Litre : double) : double
11353:   Function GalCanToLitre( const GalCan : double) : double
11354:   Function LitreToGalUk( const Litre : double) : double
11355:   Function GalUkToLitre( const GalUk : double) : double
11356:   Function KgToLb( const Kg : double) : double
11357:   Function LbToKg( const Lb : double) : double
11358:   Function KgToOz( const Kg : double) : double
11359:   Function OzToKg( const Oz : double) : double
11360:   Function CwtUsToKg( const Cwt : double) : double
11361:   Function CwtUkToKg( const Cwt : double) : double
11362:   Function KaratToKg( const Karat : double) : double
11363:   Function KgToCwtUs( const Kg : double) : double
11364:   Function KgToCwtUk( const Kg : double) : double
11365:   Function KgToKarat( const Kg : double) : double
11366:   Function KgToSton( const Kg : double) : double
11367:   Function KgToLton( const Kg : double) : double
11368:   Function StonToKg( const STon : double) : double
11369:   Function LtonToKg( const Lton : double) : double
11370:   Function QrUsToKg( const Qr : double) : double
11371:   Function QrUkToKg( const Qr : double) : double

```

```

11372: Function KgToQrUs( const Kg : double) : double
11373: Function KgToQrUk( const Kg : double) : double
11374: Function PascalToBar( const Pa : double) : double
11375: Function PascalToAt( const Pa : double) : double
11376: Function PascalToTorr( const Pa : double) : double
11377: Function BarToPascal( const Bar : double) : double
11378: Function AtToPascal( const At : double) : double
11379: Function TorrToPascal( const Torr : double) : double
11380: Function KnotToMs( const Knot : double) : double
11381: Function HpElectricToWatt( const HPE : double) : double
11382: Function HpMetricToWatt( const HpM : double) : double
11383: Function MsToKnot( const ms : double) : double
11384: Function WattToHpElectric( const W : double) : double
11385: Function WattToHpMetric( const W : double) : double
11386: function getBigPI: string; //PI of 1000 numbers
11387:
11388: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11389: begin
11390:   Function CDEExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11391:   Procedure CDCopyFile( const FileName, DestName : string)
11392:   Procedure CDMoveFile( const FileName, DestName : string)
11393:   Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11394:   Procedure CDDeleteFiles( Sender : TObject; s : string)
11395:   Function CDGetTempDir : string
11396:   Function CDGetFileSize( FileName : string) : longint
11397:   Function GetFileTime( FileName : string) : longint
11398:   Function GetShortName( FileName : string) : string
11399:   Function GetFullName( FileName : string) : string
11400:   Function WinReboot : boolean
11401:   Function WinDir : String
11402:   Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11403:   Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11404:   Function devExecutor : TdevExecutor
11405: end;
11406:
11407: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11408: begin
11409:   Procedure CheckAssociations // AssociationsCount', 'LongInt'( 7);
11410:   Procedure Associate( Index : integer)
11411:   Procedure UnAssociate( Index : integer)
11412:   Function IsAssociated( Index : integer) : boolean
11413:   Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11414:   Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11415:   Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11416:   procedure RefreshIcons;
11417:   function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11418:   function MergColor(Colors: Array of TColor): TColor;
11419:   function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11420:   procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11421:   function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11422:   function GetInverseColor(AColor: TColor): TColor;
11423:   procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11424:   procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X,Y: integer;ShadowColor: TColor);
11425:   procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11426:   Procedure GetSystemMenuFont(Font: TFont);
11427: end;
11428:
11429: //*****unit uPSI_JvHLParse;*****
11430: function IsStringConstant(const St: string): Boolean;
11431: function IsIntConstant(const St: string): Boolean;
11432: function IsRealConstant(const St: string): Boolean;
11433: function IsIdentifier(const ID: string): Boolean;
11434: function GetValue(const St: string): string;
11435: procedure ParseString(const S: string; Ss: TStrings);
11436: function IsStringConstantW(const St: WideString): Boolean;
11437: function IsIntConstantW(const St: WideString): Boolean;
11438: function IsRealConstantW(const St: WideString): Boolean;
11439: function IsIdentifierW(const ID: WideString): Boolean;
11440: function GetValueW(const St: WideString): WideString;
11441: procedure ParseStringW(const S: WideString; Ss: TStrings);
11442:
11443:
11444: //*****unit uPSI_JclMapi;*****
11445:
11446: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment : TFileName; ShowDialog : Boolean; AParentWND : HWnd) : Boolean
11447: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment : TFileName; ShowDialog : Boolean; AParentWND : HWnd) : Boolean
11448: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const AAttach:TFileName;AParentWND:HWnd):Bool
11449: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11450: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11451:
11452: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11453: begin
11454:   //Pdes_key_schedule', '^des_key_schedule // will not work
11455:   Function BuildType1Message( ADomain, AHost : String) : String
11456:   Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11457:   Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)

```

```

11458: Function FindAuthClass( AuthName : String ) : TIIdAuthenticationClass
11459: GBase64CodeTable', 'string'ABCDEFGHijklmnOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11460: GXECodeTable', 'string'+-0123456789ABCDEFGHijklmnOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11461: GUUECodeTable', 'string'!"#$%&'"()*+,-./0123456789:<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11462: end;
11463:
11464: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11465: begin
11466: ('IpAny', 'LongWord').SetUInt( $00000000 );
11467: IpLoopBack', 'LongWord').SetUInt( $7F000001 );
11468: IpBroadcast', 'LongWord').SetUInt( $FFFFFFFF );
11469: IpNone', 'LongWord').SetUInt( $FFFFFFFF );
11470: PortAny', 'LongWord( $0000 );
11471: SocketMaxConnections', 'LongInt'( 5 );
11472: TIpAddr', 'LongWord
11473: TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11474: Function HostToNetLong( HostLong : LongWord ) : LongWord
11475: Function HostToNetShort( HostShort : Word ) : Word
11476: Function NetToHostLong( NetLong : LongWord ) : LongWord
11477: Function NetToHostShort( NetShort : Word ) : Word
11478: Function StrToIp( Ip : string ) : TIpAddr
11479: Function IpToStr( Ip : TIpAddr ) : string
11480: end;
11481:
11482: (*-----*)
11483: procedure SIRegister_ALSMTPCClient(CL: TPSPascalCompiler);
11484: begin
11485: TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11486: +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11487: +'amSha1, AlsmtpClientAuthAutoSelect )
11488: TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11489: SIRegister_TAlSmtpClient(CL);
11490: end;
11491:
11492: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11493: begin
11494: 'TBitNo', 'Integer
11495: TStByteNo', 'Integer
11496: TStationNo', 'Integer
11497: TInOutNo', 'Integer
11498: TIO', '( EE, AA, NE, NA )
11499: TBitSet', 'set of TBitNo
11500: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11501: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11502: TBitAddr', 'LongInt
11503: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11504: TByteAddr', 'SmallInt
11505: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11506: Function BitAddr(aIo: TIO; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11507: Function BusbitAddr(aIo:TIO;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11508: Procedure BitAddrToValues(aBitAddr:TBitAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte;var aBitNo:TBitNo);
11509: Function BitAddrToStr( Value : TBitAddr ) : string
11510: Function StrToBitAddr( const Value : string ) : TBitAddr
11511: Function ByteAddr( aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte ) : TByteAddr
11512: Function BusByteAddr(aIo:TIO;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11513: Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte)
11514: Function ByteAddrToStr( Value : TByteAddr ) : string
11515: Function StrToByteAddr( const Value : string ) : TByteAddr
11516: Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer )
11517: Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer )
11518: Function InOutStateToStr( State : TInOutState ) : string
11519: Function MasterErrorToStr( ErrorCode : TErrorCode ) : string
11520: Function SlaveErrorToStr( ErrorCode : TErrorCode ) : string
11521: end;
11522:
11523: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11524: begin
11525: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048,
11526: +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11527: DpmiPmVector', 'Int64
11528: 'DInterval', 'LongInt'( 1000 );
11529: //'DEnabled', 'Boolean')BoolToStr( True );
11530: 'DIntFreq', 'String' if64
11531: //DMessages', 'Boolean if64);
11532: SIRegister_TwdxCustomTimer(CL);
11533: SIRegister_TwdxTimer(CL);
11534: SIRegister_TwdxRtcTimer(CL);
11535: SIRegister_TCustomIntTimer(CL);
11536: SIRegister_TIntTimer(CL);
11537: SIRegister_TRtcIntTimer(CL);
11538: Function RealNow : TDateTime
11539: Function MsToDateTIme( Millisecond : LongInt ) : TDateTime
11540: Function DateTImeToMs( Time : TDateTime ) : LongInt
11541: end;
11542:
11543: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11544: begin
11545: TIIdSyslogPRI', 'Integer

```

```

11546: TIIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11547:   +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11548:   +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11549:   +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11550:   +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )'
11551: TIIdSyslogSeverity', '( slEmergency, slAlert, slCritical, slError, slWarning, slNotice, slInformational, slDebug)'
11552: SIRegister_TIdSysLogMsgPart(CL);
11553: SIRegister_TIdSysLogMessage(CL);
11554: Function FacilityToString( AFac : TIIdSyslogFacility ) : string
11555: Function SeverityToString( ASec : TIIdSyslogSeverity ) : string
11556: Function NoToSeverity( ASev : Word ) : TIIdSyslogSeverity
11557: Function logSeverityToNo( ASev : TIIdSyslogSeverity ) : Word
11558: Function NoToFacility( AFac : Word ) : TIIdSyslogFacility
11559: Function logFacilityToNo( AFac : TIIdSyslogFacility ) : Word
11560: end;
11561;
11562: procedure SIRegister_TextUtils(CL: TPSPPascalCompiler);
11563: begin
11564:   'UWhitespace', 'String' '(?:\s*)'
11565:   Function StripSpaces( const AText : string ) : string
11566:   Function CharCount( const AText : string; Ch : Char ) : Integer
11567:   Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11568:   Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11569: end;
11570;
11571;
11572: procedure SIRegister_ExtPascalUtils(CL: TPSPPascalCompiler);
11573: begin
11574:   ExtPascalVersion', 'String' '0.9.8
11575:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11576:     +'Opera, brKonqueror, brMobileSafari )
11577:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11578:   AddTypeS('TExtProcedure', 'Procedure
11579:   Function DetermineBrowser( const UserAgentStr : string ) : TBrowser
11580:   Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11581:   Function ExtExplode( Delim : char; const S : string; Separator : char ) : TStringList
11582:   Function FirstDelimiter( const Delimiters, S : string; Offset : integer ) : integer
11583:   Function RPosEx( const Substr, Str : string; Offset : integer ) : integer
11584:   Function CountStr( const Substr, Str : string; UntilStr : string ) : integer
11585:   Function StrToJS( const S : string; UseBR : boolean ) : string
11586:   Function CaseOf( const S : string; const Cases : array of string ) : integer
11587:   Function RCaseOf( const S : string; const Cases : array of string ) : integer
11588:   Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer ) : string
11589:   Function SetPaddings(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11590:   Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11591:   Function ExtBefore( const BeforeS, AfterS, S : string ) : boolean
11592:   Function IsUpperCase( S : string ) : boolean
11593:   Function BeautifyJS( const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11594:   Function BeautifyCSS( const AStyle : string ) : string
11595:   Function LengthRegExp( Rex : string; CountAll : Boolean ) : integer
11596:   Function JSDateToDate( JSDate : string ) : TDate
11597: end;
11598;
11599: procedure SIRegister_JclShell(CL: TPSPPascalCompiler);
11600: begin
11601:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11602:   TSHDeleteOptions', 'set of TSHDeleteOption
11603:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11604:   TSHRenameOptions', 'set of TSHRenameOption
11605:   Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions ) : Boolean
11606:   Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions ) : Boolean
11607:   Function SHRenamefile( const Src, Dest : string; Options : TSHRenameOptions ) : Boolean
11608:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11609:   TEnumFolderFlags', 'set of TEnumFolderFlag
11610:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11611:   +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11612:   +'IEnumIdList; Folder : IShellFolder; end
11613:   Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11614:   Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11615:   Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11616:   Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11617:   Function GetSpecialFolderLocation( const Folder : Integer) : string
11618:   Function DisplayPropDialog( const Handle : HWND; const FileName : string ) : Boolean;
11619:   Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList ) : Boolean;
11620:   Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint ) : Boolean
11621:   Function OpenFolder( const Path : string; Parent : HWND ) : Boolean
11622:   Function OpenSpecialFolder( FolderID : Integer; Parent : HWND ) : Boolean
11623:   Function SHReallocMem( var P : Pointer; Count : Integer ) : Boolean
11624:   Function SHAllocMem( out P : Pointer; Count : Integer ) : Boolean
11625:   Function SHGetMem( var P : Pointer; Count : Integer ) : Boolean
11626:   Function SHFreeMem( var P : Pointer ) : Boolean
11627:   Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder ) : PItemIdList
11628:   Function PathToPidl( const Path : string; Folder : IShellFolder ) : PItemIdList
11629:   Function PathToPidlBind( const FileName : string; out Folder : IShellFolder ) : PItemIdList
11630:   Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11631:   Function PidlCompare( const Pid1, Pid2 : PItemIdList ) : Boolean
11632:   Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList ) : Boolean
11633:   Function PidlFree( var IdList : PItemIdList ) : Boolean
11634:   Function PidlGetDepth( const Pidl : PItemIdList ) : Integer

```

```

11635: Function PidlGetLength( const Pidl : PIItemIdList ) : Integer
11636: Function PidlGetNext( const Pidl : PIItemIdList ) : PIItemIdList
11637: Function PidlToPath( IdList : PIItemIdList ) : string
11638: Function StrRetFreeMem( StrRet : TStrRet ) : Boolean
11639: Function StrRetToString( IdList : PIItemIdList; StrRet : TStrRet; Free : Boolean ) : string
11640: PShellLink', '^TShellLink // will not work
11641: TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11642: +'ingDirectory : string; IdList : PIItemIDList; Target : string; Description '
11643: +' : string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11644: Procedure ShellLinkFree( var Link : TShellLink )
11645: Function ShellLinkResolve( const FileName : string; var Link : TShellLink ) : HRESULT
11646: Function ShellLinkCreate( const Link : TShellLink; const FileName : string ) : HRESULT
11647: Function ShellLinkCreateSystem( const Link:TShellLink;const Folder:Integer; const FileName:string ):HRESULT;
11648: Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon ) : Boolean
11649: Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo ) : Boolean
11650: Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal ) : HICON
11651: Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean ) : Boolean
11652: Function OverlayIconShortCut( var Large, Small : HICON ) : Boolean
11653: Function OverlayIconShared( var Large, Small : HICON ) : Boolean
11654: Function SHGetFileInfoTip( const Folder : IShellFolder; Item : PIItemIdList ) : string
11655: Function ShellExecEx( const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int ):Bool;
11656: Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int ):Bool;
11657: Function ShellExecAndWait( const FileName:string;const Params: string;const Verb:string;CmdShow:Int ):Bool;
11658: Function ShellOpenAs( const FileName : string ) : Boolean
11659: Function ShellRasodial( const EntryName : string ) : Boolean
11660: Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer ):Boolean
11661: Function GetFileNameIcon( const FileName : string; Flags : Cardinal ) : HICON
11662: TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11663: Function GetfileExeType( const FileName : TfileName ) : TJclFileExeType
11664: Function ShellFindExecutable( const FileName, DefaultDir : string ) : string
11665: Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD )
11666: Function OemKeyScan( wOemChar : Word ) : DWORD
11667: Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD )
11668: end;
11669:
11670: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11671: begin
11672: xmlVersion', 'String '1.0 FindClass('TOBJECT'),'Exml
11673: //Function xmlValidChar( const Ch : AnsiChar ) : Boolean
11674: Function xmlValidChar1( const Ch : UCS4Char ) : Boolean;
11675: Function xmlValidChar2( const Ch : WideChar ) : Boolean;
11676: Function xmlIsSpaceChar( const Ch : WideChar ) : Boolean
11677: Function xmlIsLetter( const Ch : WideChar ) : Boolean
11678: Function xmlIsDigit( const Ch : WideChar ) : Boolean
11679: Function xmlIsNameStartChar( const Ch : WideChar ) : Boolean
11680: Function xmlIsNameChar( const Ch : WideChar ) : Boolean
11681: Function xmlIsPubidChar( const Ch : WideChar ) : Boolean
11682: Function xmlValidName( const Text : UnicodeString ) : Boolean
11683: //xmlSpace', 'Char #$20 or #$9 or #$D or #$A';
11684: //Function xmlSkipSpace( var P : PWideChar ) : Boolean
11685: //Function xmlSkipBq( var P : PWideChar ) : Boolean
11686: //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString ) : Boolean
11687: //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer )
11688: : TUnicodeCodecClass
11689: Function xmlResolveEntityReference( const RefName : UnicodeString ) : WideChar
11690: Function xmlTag( const Tag : UnicodeString ) : UnicodeString
11691: Function xmlEndTag( const Tag : UnicodeString ) : UnicodeString
11692: Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString ) : UnicodeString
11693: Procedure xmlSafeTextInPlace( var Txt : UnicodeString )
11694: Function xmlSafeText( const Txt : UnicodeString ) : UnicodeString
11695: Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer ) : UnicodeString
11696: Function xmlTabIndent( const IndentLevel : Integer ) : UnicodeString
11697: Function xmlComment( const Comment : UnicodeString ) : UnicodeString
11698: Procedure SelfTestcXMLFunctions
11699: end;
11700:
11701: (*-----*)
11702: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11703: begin
11704: Function AWaitCursor : IUnknown
11705: Function ChangeCursor( NewCursor : TCursor ) : IUnknown
11706: Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean )
11707: Function YesNo( const ACaption, AMsg : string ) : boolean
11708: Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings )
11709: Function GetBorlandLibPath( Version : integer; ForDelphi : boolean ) : string
11710: Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean ) : string
11711: Procedure GetPathlist( Version : integer; ForDelphi : boolean; Strings : TStrings )
11712: Procedure GetSystemPaths( Strings : TStrings )
11713: Procedure MakeEditNumeric( EditHandle : integer )
11714: end;
11715:
11716: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11717: begin
11718: AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11719: 'BI_YUY2','LongWord( $3259559 );
11720: 'BI_UYVY','LongWord').SetUInt( $59565955 );
11721: 'BI_BTYUV','LongWord').SetUInt( $50313459 );
11722: 'BI_YVU9','LongWord').SetUInt( $39555659 );

```

```

11723: 'BI_YUV12','LongWord( $30323449);
11724: 'BI_Y8','LongWord').SetUInt( $20203859);
11725: 'BI_Y211','LongWord').SetUInt( $31313259);
11726: Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec;
11727: Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11728: end;
11729:
11730: (*-----*)
11731: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11732: begin
11733: 'WM_USER','LongWord').SetUInt( $0400);
11734: 'WM_CAP_START','LongWord').SetUInt($0400);
11735: 'WM_CAP_END','longword').SetUInt($0400+85);
11736: //WM_CAP_START+ 85
11737: // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11738: Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11739: Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11740: Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11741: Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11742: Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11743: Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11744: Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11745: Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11746: Function capGetUserData( hwnd : THandle) : LongInt
11747: Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11748: Function capDriverDisconnect( hwnd : THandle) : LongInt
11749: Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11750: Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11751: Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11752: Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11753: Function capfileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11754: Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11755: Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11756: Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11757: Function capfileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11758: Function capEditCopy( hwnd : THandle) : LongInt
11759: Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11760: Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11761: Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11762: Function capDlgVideoFormat( hwnd : THandle) : LongInt
11763: Function capDlgVideoSource( hwnd : THandle) : LongInt
11764: Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11765: Function capDlgVideoCompression( hwnd : THandle) : LongInt
11766: Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11767: Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11768: Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11769: Function capPreview( hwnd : THandle; f : Word) : LongInt
11770: Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11771: Function capOverlay( hwnd : THandle; f : Word) : LongInt
11772: Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11773: Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11774: Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11775: Function capGrabFrame( hwnd : THandle) : LongInt
11776: Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11777: Function capCaptureSequence( hwnd : THandle) : LongInt
11778: Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11779: Function capCaptureStop( hwnd : THandle) : LongInt
11780: Function capCaptureAbort( hwnd : THandle) : LongInt
11781: Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11782: Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11783: Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11784: Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11785: Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11786: Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11787: Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11788: Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11789: Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11790: Function capPalettePaste( hwnd : THandle) : LongInt
11791: Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11792: Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11793: //PCapDriverCaps', '^TCapDriverCaps // will not work
11794: TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11795: +' ; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11796: +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hvid'
11797: +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11798: //PCapStatus', '^TCapStatus // will not work
11799: TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11800: +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOINT'
11801: +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11802: +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11803: +'rrentWaveSamples : DWORD; dwCurrentTimeLapsedMS : DWORD; hPalCurrent : HP'
11804: +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD; '
11805: +' wNumAudioAllocated : WORD; end
11806: //PCaptureParms', '^TCaptureParms // will not work
11807: TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11808: +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11809: +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11810: +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11811: +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'

```

```

11812: +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11813: +'wMCISearchBar : WORD; dwMCISearchTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11814: +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11815: +'he : BOOL; AVStreamMaster : WORD; end
11816: // PCapInfoChunk, '^TCapInfoChunk // will not work
11817: //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11818: 'CONTROLCALLBACK_PREROLL', 'LongInt'( 1);
11819: 'CONTROLCALLBACK_CAPTURING', 'LongInt'( 2);
11820: Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
: Integer; hwndParent : THandle; nID : Integer ) : THandle
11821: Function
capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11822: 'IDS_CAP_BEGIN', 'LongInt'( 300);
11823: 'IDS_CAP_END', 'LongInt'( 301);
11824: 'IDS_CAP_INFO', 'LongInt'( 401);
11825: 'IDS_CAP_OUTOFMEM', 'LongInt'( 402);
11826: 'IDS_CAP_FILEEXISTS', 'LongInt'( 403);
11827: 'IDS_CAP_ERRORPALOPEN', 'LongInt'( 404);
11828: 'IDS_CAP_ERRORPALSOLVE', 'LongInt'( 405);
11829: 'IDS_CAP_ERRORDIBSAVE', 'LongInt'( 406);
11830: 'IDS_CAP_DEFAVIEXT', 'LongInt'( 407);
11831: 'IDS_CAP_DEFPALEXT', 'LongInt'( 408);
11832: 'IDS_CAP_CANTOPEN', 'LongInt'( 409);
11833: 'IDS_CAP_SEQ_MSGSTART', 'LongInt'( 410);
11834: 'IDS_CAP_SEQ_MSGSTOP', 'LongInt'( 411);
11835: 'IDS_CAP_VIDEEDITERR', 'LongInt'( 412);
11836: 'IDS_CAP_READONLYFILE', 'LongInt'( 413);
11837: 'IDS_CAP_WRITEERROR', 'LongInt'( 414);
11838: 'IDS_CAP_NODISKSPACE', 'LongInt'( 415);
11839: 'IDS_CAP_SETFILESIZE', 'LongInt'( 416);
11840: 'IDS_CAP_SAVEASPERCENT', 'LongInt'( 417);
11841: 'IDS_CAP_DRIVER_ERROR', 'LongInt'( 418);
11842: 'IDS_CAP_WAVE_OPEN_ERROR', 'LongInt'( 419);
11843: 'IDS_CAP_WAVE_ALLOC_ERROR', 'LongInt'( 420);
11844: 'IDS_CAP_WAVE_PREPARE_ERROR', 'LongInt'( 421);
11845: 'IDS_CAP_WAVE_ADD_ERROR', 'LongInt'( 422);
11846: 'IDS_CAP_WAVE_SIZE_ERROR', 'LongInt'( 423);
11847: 'IDS_CAP_VIDEO_OPEN_ERROR', 'LongInt'( 424);
11848: 'IDS_CAP_VIDEO_ALLOC_ERROR', 'LongInt'( 425);
11849: 'IDS_CAP_VIDEO_PREPARE_ERROR', 'LongInt'( 426);
11850: 'IDS_CAP_VIDEO_ADD_ERROR', 'LongInt'( 427);
11851: 'IDS_CAP_VIDEO_SIZE_ERROR', 'LongInt'( 428);
11852: 'IDS_CAP_FILE_OPEN_ERROR', 'LongInt'( 429);
11853: 'IDS_CAP_FILE_WRITE_ERROR', 'LongInt'( 430);
11854: 'IDS_CAP_RECORDING_ERROR', 'LongInt'( 431);
11855: 'IDS_CAP_RECORDING_ERROR2', 'LongInt'( 432);
11856: 'IDS_CAP_AVI_INIT_ERROR', 'LongInt'( 433);
11857: 'IDS_CAP_NO_FRAME_CAP_ERROR', 'LongInt'( 434);
11858: 'IDS_CAP_NO_PALETTE_WARN', 'LongInt'( 435);
11859: 'IDS_CAP_MCI_CONTROL_ERROR', 'LongInt'( 436);
11860: 'IDS_CAP_MCI_CANT_STEP_ERROR', 'LongInt'( 437);
11861: 'IDS_CAP_NO_AUDIO_CAP_ERROR', 'LongInt'( 438);
11862: 'IDS_CAP_AVI_DRAWDIB_ERROR', 'LongInt'( 439);
11863: 'IDS_CAP_COMPRESSOR_ERROR', 'LongInt'( 440);
11864: 'IDS_CAP_AUDIO_DROP_ERROR', 'LongInt'( 441);
11865: 'IDS_CAP_STAT_LIVE_MODE', 'LongInt'( 500);
11866: 'IDS_CAP_STAT_OVERLAY_MODE', 'LongInt'( 501);
11867: 'IDS_CAP_STAT_CAP_INIT', 'LongInt'( 502);
11868: 'IDS_CAP_STAT_CAP_FINI', 'LongInt'( 503);
11869: 'IDS_CAP_STAT_PALETTE_BUILD', 'LongInt'( 504);
11870: 'IDS_CAP_STAT_OPTPAL_BUILD', 'LongInt'( 505);
11871: 'IDS_CAP_STAT_I_FRAMES', 'LongInt'( 506);
11872: 'IDS_CAP_STAT_L_FRAMES', 'LongInt'( 507);
11873: 'IDS_CAP_STAT_CAP_L_FRAMES', 'LongInt'( 508);
11874: 'IDS_CAP_STAT_CAP_AUDIO', 'LongInt'( 509);
11875: 'IDS_CAP_STAT_VIDEOCURRENT', 'LongInt'( 510);
11876: 'IDS_CAP_STAT_VIDEOAUDIO', 'LongInt'( 511);
11877: 'IDS_CAP_STAT_VIDEOONLY', 'LongInt'( 512);
11878: 'IDS_CAP_STAT_FRAMESDROPPED', 'LongInt'( 513);
11879: 'AVICAP32', 'String' 'AVICAP32.dll
11880: end;
11881:
11882: procedure SIRegister_ALFcnnMisc(CL: TPSPascalCompiler);
11883: begin
11884: Function AlBoolToInt( Value : Boolean ) : Integer
11885: Function ALMediumPos( LTotal, LBorder, LObject : integer ) : Integer
11886: Function AlIsValidEmail( const Value : AnsiString ) : boolean
11887: Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime ) : TdateTime
11888: Function ALInc( var x : integer; Count : integer ) : Integer
11889: function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11890: function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11891: procedure ALSaveStringToFile(Str: AnsiString; filename: AnsiString);
11892: Function AlIsInteger(const S: AnsiString): Boolean;
11893: function AlIsDecimal(const S: AnsiString): boolean;
11894: Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11895: function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11896: function ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''): AnsiString;
11897: function ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11898: function ALUTF8removeBOM(const S: AnsiString): AnsiString;

```

```

11899: Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11900: Function ALRandomStr(const aLength: Longint): AnsiString;
11901: Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11902: Function ALRandomStrU(const aLength: Longint): String;
11903: end;
11904:
11905: procedure SIRегистер_ALJSONDoc(CL: TPSPPascalCompiler);
11906: begin
11907:   Procedure ALJSONToTStrings(const AJsonStr: AnsiString; aLst: TALStrings; const aNullStr: AnsiString; const
11908:     aTrueStr: AnsiString; const aFalseStr: AnsiString)
11909:   end;
11910:   procedure SIRегистер_ALWindows(CL: TPSPPascalCompiler);
11911:   begin
11912:     _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11913:       +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11914:       +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11915:       +'; ullAvailExtendedVirtual : Int64; end
11916:     TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11917:     Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX ) : BOOL
11918:     Function ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG ) : LONGLONG
11919:     'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ) );
11920:     'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2 );
11921:     'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1 );
11922:     'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8 );
11923:     'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4 );
11924:   end;
11925:
11926:   procedure SIRегистер_IPCThrd(CL: TPSPPascalCompiler);
11927:   begin
11928:     SIRегистер_THandledObject(CL);
11929:     SIRегистер_TEvent(CL);
11930:     SIRегистер_TMutex(CL);
11931:     SIRегистер_TSharedMem(CL);
11932:     'TRACE_BUF_SIZE','LongInt'( 200 * 1024 );
11933:     'TRACE_BUFFER','String 'TRACE_BUFFER
11934:     'TRACE_MUTEX','String 'TRACE_MUTEX
11935:     //PTTraceEntry', 'TTTraceEntry // will not work
11936:     SIRегистер_TIPCTracer(CL);
11937:     'MAX_CLIENTS','LongInt'( 6 );
11938:     'IPC TIMEOUT','LongInt'( 2000 );
11939:     'IPCBUFFER_NAME','String 'BUFFER_NAME
11940:     'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11941:     'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11942:     'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11943:     'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11944:     'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11945:     'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11946:     FindClass('TOBJECT'), EMonitorActive
11947:     FindClass('TOBJECT'), TIPTThread
11948:     TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11949:       +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11950:       +'ach, evClientSwitch, evClientSignal, evClientExit )
11951:     TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11952:     TClientFlags', 'set of TClientFlag
11953:     //PEventData', 'TEventData // will not work
11954:     TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11955:       +'lag; Flags : TClientFlags; end
11956:     TConnectEvent', 'Procedure ( Sender : TIPTThread; Connecting : Boolean )
11957:     TDirUpdateEvent', 'Procedure ( Sender : TIPTThread )
11958:     TIPNotifyEvent', 'Procedure ( Sender : TIPTThread; Data : TEventData )
11959:     //PIPCEventInfo', '^TIPCEventInfo // will not work
11960:     TIPCEventInfo', 'record FID:Integer; FKind:TEventKind; FData:TEventData;end
11961:     SIRегистер_TIPCEvent(CL);
11962:     //PClientDirRecords', '^TClientDirRecords // will not work
11963:     SIRегистер_TClientDirectory(CL);
11964:     TIPOState', '( stInactive, stDisconnected, stConnected )
11965:     SIRегистер_TIPTThread(CL);
11966:     SIRегистер_TIPCMonitor(CL);
11967:     SIRегистер_TIPCCClient(CL);
11968:     Function IsMonitorRunning( var Hndl : THandle ) : Boolean
11969:   end;
11970:
11971: (*-----*)
11972: procedure SIRегистер_ALGSMComm(CL: TPSPPascalCompiler);
11973: begin
11974:   SIRегистер_TALGSMComm(CL);
11975:   Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString ) : AnsiString
11976:   Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
11977:     AMessage:AnsiString);
11978:   Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString ) : AnsiString
11979:   Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
11980:     UseGreekAlphabet:Bool):Widestring;
11981:   function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11982:   procedure SIRегистер_ALHttpCommon(CL: TPSPPascalCompiler);
11983:   begin
11984:     TALHTTPPropertyChangeEvent', 'Procedure(sender:Tobject;const PropertyIndex:Integer;

```

```

11985: TALHTTPProtocolVersion', '( HTTPPpv_1_0, HTTPPpv_1_1 )
11986: TALHTTPMethod', '(HTTPPmt_Get,HTTPPmt_Post,HTTPPmt_Head,HTTPPmt_Trace,HTTPPmt_Put,HTTPPmt_Delete);
11987: TIInternetScheme', 'integer
11988: TALIPv6Binary', 'array[1..16] of Char;
11989: // TALIPv6Binary = array[1..16] of ansiChar;
11990: // TIInternetScheme = Integer;
11991: SIRегистer_TALHTTPRequestHeader(CL);
11992: SIRегистer_TALHTTPCookie(CL);
11993: SIRегистer_TALHTTPCookieCollection(CL);
11994: SIRегистer_TALHTTPResponseHeader(CL);
11995: Function ALHTTPDecode( const AStr : AnsiString ) : AnsiString
11996: Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings )
11997: // Procedure ALEExtractHTTPFields( Separators,WhiteSpace,Quotes:TSysCharSet,
Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean);
11998: // Procedure ALEExtractHeaderFields( Separators,WhiteSpace,Quotes : TSysCharSet; Content : PAnsiChar;
Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11999: // Procedure ALEExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
12000: Function AlRemoveShemeFromUrl( aUrl : AnsiString ) : ansiString
12001: Function AlExtractShemeFromUrl( aUrl : AnsiString ) : TIInternetScheme
12002: Function AlExtractHostNameFromUrl( aUrl : AnsiString ) : AnsiString
12003: Function AlExtractDomainNameFromUrl( aUrl : AnsiString ) : AnsiString
12004: Function AlExtractUrlPathFromUrl( aUrl : AnsiString ) : AnsiString
12005: Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
ExtraInfo : AnsiString; var PortNumber : integer ) : Boolean;
12006: Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
Anchor : AnsiString; Query : TALStrings; var PortNumber : integer ) : Boolean;
12007: Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
12008: Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString ) : AnsiString;
12009: Function AlRemoveAnchorFromUrl1( aUrl : AnsiString ) : AnsiString;
12010: Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString ) : AnsiString;
12011: Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
12012: Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
12013: Function ALDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
12014: Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime ) : Boolean
12015: Function ALRfc822StrToGMTDateTime( const s : AnsiString ) : TDateTime
12016: Function ALTryIPv4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal ) : Boolean
12017: Function ALIPv4StrToNumeric( aIPv4 : ansiString ) : Cardinal
12018: Function ALNumericToIPv4Str( aIPv4 : Cardinal ) : ansiString
12019: Function ALZeroIpV6 : TALIPv6Binary
12020: Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary ) : Boolean
12021: Function ALIPv6StrTobinary( aIPv6 : ansiString ) : TALIPv6Binary
12022: Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary ) : ansiString
12023: Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString ) : TALIPv6Binary
12024: end;
12025:
12026: procedure SIRегистer_ALFcnsHTML(CL: TPSPascalCompiler);
12027: begin
12028:   Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
DecodeHTMLText:Bool;
12029:   Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
12030:   Function ALXMLCDataElementEncode( Src : AnsiString ) : AnsiString
12031:   Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean ) : AnsiString
12032:   Function ALUTF8XMLTextElementDecode( const Src : AnsiString ) : AnsiString
12033:   Function ALUTF8HTMLDecode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
useNumRef:bool):AnsiString);
12034:   Function ALUTF8HTMLDecode( const Src : AnsiString ) : AnsiString
12035:   Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean ) : AnsiString
12036:   Function ALUTF8JavascriptDecode( const Src : AnsiString ) : AnsiString
12037:   Procedure ALHideHTMLUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12038:   Procedure ALCompactHTMLTagParams( TagParams : TALStrings )
12039: end;
12040:
12041: procedure SIRегистer_ALInternetMessageCommon(CL: TPSPascalCompiler);
12042: begin
12043:   SIRегистer_TALEMailHeader(CL);
12044:   SIRегистer_TALNewsArticleHeader(CL);
12045:   Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
decodeRealName:Bool):AnsiString;
12046:   Function AlExtractEmailAddress( FriendlyEmail : AnsiString ) : AnsiString
12047:   Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString ) : AnsiString
12048:   Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString ) : AnsiString
12049:   Function AlGenerateInternetMessageID : AnsiString;
12050:   Function ALGenerateInternetMessageID1( ahostname : AnsiString ) : AnsiString;
12051:   Function ALDecodeQuotedPrintableString( src : AnsiString ) : AnsiString
12052:   Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12053: end;
12054:
12055: (*-----*)
12056: procedure SIRегистer_ALFcnsWinSock(CL: TPSPascalCompiler);
12057: begin
12058:   Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12059:   Function ALIPAddrToName( IPAddr : AnsiString ) : AnsiString
12060:   Function ALgetLocalIPs : TALStrings
12061:   Function ALgetLocalHostName : AnsiString
12062: end;
12063:
12064: procedure SIRегистer_ALFcnsCGI(CL: TPSPascalCompiler);

```

```

12065: begin
12066:   Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
TALWebRequest;ServerVariables:TALStrings);
12067:   Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
TALStrings; ScriptName , ScriptFileName : AnsiString; Url : AnsiString);
12068:   Procedure AlCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12069:   Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
ScriptFileName:AnsiString;Url:Ansistr;
12070:   Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
ScriptName , ScriptFileName : AnsiString; Url : AnsiString);
12071:   Procedure AlCGIEexec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
: Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12072:   Procedure AlCGIEexec1(ScriptName,ScriptFileName , Url, X_REWRITE_URL , InterpreterFilename:AnsiString;
WebRequest : TALIsapiRequest;
overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;
12073:   +'overloadedRequestContentStream:Tstream;var
ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12074:   Procedure AlCGIEexec2(ScriptName,ScriptFileName , Url,X_REWRITE_URL,
InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
ResponseHeader : TALHTTPResponseHeader);
12075: end;
12076:
12077: procedure SIRegister_ALFcncExecute(CL: TPSPascalCompiler);
12078: begin
12079:   TStartupInfoA' , 'TStartupInfo
12080:   'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12081:   SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12082:   SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12083:   SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12084:   SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12085:   SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12086:   SE_TCB_NAME','String 'SeTcbPrivilege
12087:   SE_SECURITY_NAME','String 'SeSecurityPrivilege
12088:   SE TAKE OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12089:   SE LOAD DRIVER_NAME','String 'SeLoadDriverPrivilege
12090:   SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12091:   SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12092:   SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12093:   SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12094:   SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12095:   SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12096:   SE_BACKUP_NAME','String 'SeBackupPrivilege
12097:   SE_RESTORE_NAME','String 'SeRestorePrivilege
12098:   SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12099:   SE_DEBUG_NAME','String 'SeDebugPrivilege
12100:   SE_AUDIT_NAME','String 'SeAuditPrivilege
12101:   SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12102:   SE_CHANGE_NOTIFY_NAME','string 'SeChangeNotifyPrivilege
12103:   SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12104:   SE_UNDOCK_NAME','String 'SeUndockPrivilege
12105:   SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12106:   SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12107:   SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12108:   Function AlGetEnvironmentString : AnsiString
12109:   Function ALWinExec32(const FileName,CurrentDir,
Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12110:   Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12111:   Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12112:   Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12113:   Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12114: end;
12115:
12116: procedure SIRegister_ALFcncFile(CL: TPSPascalCompiler);
12117: begin
12118:   Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12119:   Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12120:   Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12121:   Function ALGetModuleName : ansistring
12122:   Function ALGetModuleFileNameWithoutExtension : ansistring
12123:   Function ALGetModulePath : ansiString
12124:   Function AlGetFileSize( const AFileName : ansistring) : int64
12125:   Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12126:   Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12127:   Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12128:   Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12129:   Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12130:   Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12131:   Function ALFileExists( const Path : ansiString) : boolean
12132:   Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12133:   Function ALCreateDir( const Dir : Ansistring) : Boolean
12134:   Function ALRemoveDir( const Dir : Ansistring) : Boolean
12135:   Function ALDeleteFile( const FileName : Ansistring) : Boolean
12136:   Function ALRenameFile( const OldName , NewName : ansistring) : Boolean
12137: end;
12138:
12139: procedure SIRegister_ALFcncMime(CL: TPSPascalCompiler);

```

```

12140: begin
12141:   NativeInt', 'Integer
12142:   NativeUInt', 'Cardinal
12143:   Function ALMimeBase64EncodeString( const S : AnsiString ) : AnsiString
12144:   Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString ) : AnsiString
12145:   Function ALMimeBase64DecodeString( const S : AnsiString ) : AnsiString
12146:   Function ALMimeBase64EncodedSize( const InputSize : NativeInt ) : NativeInt
12147:   Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt ) : NativeInt
12148:   Function ALMimeBase64DecodedSize( const InputSize : NativeInt ) : NativeInt
12149:   Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12150:   Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12151:   Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12152:   Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt ) : NativeInt;
12153:   Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; '
12154:   + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12155:   Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12156:   Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
OutputBuf:TByteDynArray);
12157:   Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12158:   Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12159:   Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray):NativeInt;
12160:   Function ALMimeBase64DecodePartial11(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer : TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12161:   Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
ByteBufferSpace:Cardinal):NativeInt;
12162:   Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12163:   Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12164:   Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12165:   Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12166:   Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12167:   Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12168:   'cALMimeBase64_ENCODED_LINE_BREAK', 'LongInt'( 76 );
12169:   'cALMimeBase64_DECODED_LINE_BREAK', 'LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3 );
12170:   'cALMimeBase64_BUFFER_SIZE', 'LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4 );
12171:   Procedure ALFillMimeTypeByExtList( AMIMEList : TALStrings)
12172:   Procedure ALFillExtByMimeTypeList( AMIMEList : TALStrings)
12173:   Function ALGetDefaultFileExtFromMimeType( aContentType : AnsiString ) : AnsiString
12174:   Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString ) : AnsiString
12175: end;
12176:
12177: procedure SIRegister_ALXmlDoc(CL: TPPascalCompiler);
12178: begin
12179:   'cALXMLNodeMaxListSize','LongInt'( Maxint div 16 );
12180:   FindClass( 'TOBJECT' ),'TALXMLNode
12181:   FindClass( 'TOBJECT' ),'TALXMLNodeList
12182:   FindClass( 'TOBJECT' ),'TALXMLDocument
12183:   TALXMLParseProcessingInstructionEvent', 'Procedure ( Sender:TObject; const Target,Data:AnsiString )
12184:   TALXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString )
12185:   TALXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12186:   +'nst Name : AnsiString; const Attributes : TALStrings )
12187:   TALXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString )
12188:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12189:   +'ntCDATA, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12190:   +'ntDocType, ntDocFragment, ntNotation )
12191:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12192:   TALXMLDocOptions', 'set of TALXMLDocOption
12193:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12194:   TALXMLParseOptions', 'set of TALXMLParseOption
12195:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12196:   PALPointerXMLNodeList', '^PALPointerXMLNodeList // will not work
12197:   SIRegister_EALXMLDocError(CL);
12198:   SIRegister_TALXMLNodeList(CL);
12199:   SIRegister_TALXMLNode(CL);
12200:   SIRegister_TALXmlElementNode(CL);
12201:   SIRegister_TALXmlAttributeNode(CL);
12202:   SIRegister_TALXmlTextNode(CL);
12203:   SIRegister_TALXmlDocumentNode(CL);
12204:   SIRegister_TALXmlCommentNode(CL);
12205:   SIRegister_TALXmlProcessingInstrNode(CL);
12206:   SIRegister_TALXmlCDataNode(CL);
12207:   SIRegister_TALXmlEntityRefNode(CL);
12208:   SIRegister_TALXmlEntityNode(CL);
12209:   SIRegister_TALXmlDocTypeNode(CL);
12210:   SIRegister_TALXmlDocFragmentNode(CL);
12211:   SIRegister_TALXmlNotationNode(CL);
12212:   SIRegister_TALXMLDocument(CL);
12213:   cALXMLUTF8EncodingStr', 'String 'UTF-8
12214:   cALxmlUTF8HeaderStr', 'String <?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>' +#13#10';
12215:   CALNSDelim', 'String ': '
12216:   CALXML', 'String 'xml

```

```

12217: CALVersion', 'String 'version
12218: CALEncoding', 'String 'encoding
12219: CALStandalone', 'String 'standalone
12220: CALDefaultNodeIndent', 'String '
12221: CALXmlDocument', 'String 'DOCUMENT
12222: Function ALCreateEmptyXMLDocument( const Rootname : AnsiString ) : TalXMLDocument
12223: Procedure ALClearXMLDocument( const rootname:AnsiString;xmldoc:TalXMLDocument;const
EncodingStr:AnsiString);
12224: Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildnodeName,
ChildnodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12225: Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
ChildnodeName, ChildnodeValue : AnsiString; const Recurse : Boolean ) : TalxmlNode
12226: Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
Recurse: Boolean):TalxmlNode
12227: Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
AttributeName,AttributeValue : AnsiString; const Recurse : Boolean ) : TalxmlNode
12228: Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString ) :
AnsiString
12229: end;
12230:
12231: procedure SIRegister_TeCanvas(CL: TPSPPascalCompiler);
12232: //based on TEEProc, TeCanvas, TEEEngine, TChart
12233: begin
12234:   'TeePiStep','Double').setExtended( Pi / 180.0);
12235:   'TeeDefaultPerspective','LongInt'( 100 );
12236:   'TeeMinAngle','LongInt'( 270 );
12237:   'teeclMoneyGreen','LongWord').SetUInt( TColor( $C0DCC0 ) );
12238:   'teeclSkyBlue','LongWord').SetUInt( TColor( $F0CAA6 ) );
12239:   'teeclCream','LongWord( TColor( $F0FBFF ) );
12240:   'teeclMedGray','LongWord').SetUInt( TColor( $A4A0A0 ) );
12241:   'teeclMoneyGreen','LongWord').SetUInt( TColor( $C0DCC0 ) );
12242:   'teeclSkyBlue','LongWord').SetUInt( TColor( $F0CAA6 ) );
12243:   'teeclCream','LongWord').SetUInt( TColor( $F0FBFF ) );
12244:   'teeclMedGray','LongWord').SetUInt( TColor( $A4A0A0 ) );
12245:   'TA_LEFT','LongInt'( 0 );
12246:   'TA_RIGHT','LongInt'( 2 );
12247:   'TA_CENTER','LongInt'( 6 );
12248:   'TA_TOP','LongInt'( 0 );
12249:   'TA_BOTTOM','LongInt'( 8 );
12250:   'teePATCOPY','LongInt'( 0 );
12251:   'NumCirclePoints','Longint'( 64 );
12252:   'teeDEFAULT_CHARSET','LongInt'( 1 );
12253:   'teeANTIALIASED_QUALITY','LongInt'( 4 );
12254:   'TA_LEFT','LongInt'( 0 );
12255:   'bs_Solid','LongInt'( 0 );
12256:   'teepf24Bit','LongInt'( 0 );
12257:   'teepfDevice ','LongInt'( 1 );
12258:   'CM_MOUSELEAVE','LongInt'( 10000 );
12259:   'CM_SYSCOLORCHANGE','LongInt'( 10001 );
12260:   'DC_BRUSH','LongInt'( 18 );
12261:   'DC_PEN','LongInt'( 19 );
12262:   teeCOLORREF', 'LongWord
12263:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12264: //TNotifyEvent', 'Procedure ( Sender : TObject )
12265: SIRegister_TFilterRegion(CL);
12266: SIRegister_IFormCreator(CL);
12267: SIRegister_TTeeFilter(CL);
12268: //TFilterClass', 'class of TTeeFilter
12269: SIRegister_TFilterItems(CL);
12270: SIRegister_TConvolveFilter(CL);
12271: SIRegister_TBlurFilter(CL);
12272: SIRegister_TTeePicture(CL);
12273: TPenEndStyle', '( esRound, esSquare, esFlat )
12274: SIRegister_TChartPen(CL);
12275: SIRegister_TChartHiddenPen(CL);
12276: SIRegister_TDottedGrayPen(CL);
12277: SIRegister_TDGrayPen(CL);
12278: SIRegister_TWhitePen(CL);
12279: SIRegister_TChartBrush(CL);
12280: TTeeView3DSScrolled', 'Procedure ( IsHoriz : Boolean)
12281: TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12282: SIRegister_TVview3DOptions(CL);
12283: FindClass('TOBJECT'),TTeeCanvas
12284: TTeeTransparency', 'Integer
12285: SIRegister_TTeeBlend(CL);
12286: FindClass('TOBJECT'),TCanvas3D
12287: SIRegister_TTeeShadow(CL);
12288: teeTGradientDirection', '(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12289: FindClass('TOBJECT'),TSubGradient
12290: SIRegister_TCustomTeeGradient(CL);
12291: SIRegister_TSubGradient(CL);
12292: SIRegister_TTeeGradient(CL);
12293: SIRegister_TTeeFontGradient(CL);
12294: SIRegister_TTeeFont(CL);
12295: TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12296: TCanvasTextAlign', 'Integer
12297: TTeeCanvasHandle', 'HDC
12298: SIRegister_TTeeCanvas(CL);

```

```

12299: TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12300: SIRegister_TFloatXYZ(CL);
12301: TPoint3D', 'record x : integer; y : integer; z : Integer; end
12302: TRGB', 'record blue: byte; green: byte; red: byte; end
12303: {TRGB=packed record
12304:   Blue : Byte;
12305:   Green : Byte;
12306:   Red : Byte;
12307: //$$IFDEF CLX //Alpha : Byte; // Linux end;}
12308:
12309: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 :
12310:   +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12311: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12312: TCanvas3DPlane', '( cpX, cpY, cpZ )
12313: //IInterface', 'IUnknown
12314: SIRegister_TCanvas3D(CL);
12315: SIRegister_TTeeCanvas3D(CL);
12316: TTrianglePoints', 'Array[0..2] of TPoint;
12317: TFourPoints', 'Array[0..3] of TPoint;
12318: Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12319: Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12320: Function Point3D( const x, y, z : Integer) : TPoint3D
12321: Procedure SwapDouble( var a, b : Double)
12322: Procedure SwapInteger( var a, b : Integer)
12323: Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12324: Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12325: Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12326: Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12327: Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12328: Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12329: Procedure UnClipCanvas( ACanvas : TCanvas)
12330: Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12331: Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12332: Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12333: 'TeeCharForHeight','String 'W
12334: 'DarkerColorQuantity','Byte').SetUInt( 128);
12335: 'DarkColorQuantity','Byte').SetUInt( 64);
12336: TButtonGetColorProc', 'Function : TColor
12337: SIRegister_TTeeButton(CL);
12338: SIRegister_TButtonColor(CL);
12339: SIRegister_TComboFlat(CL);
12340: Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12341: Function TeePoint( const ax, ay : Integer) : TPoint
12342: Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12343: Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12344: Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12345: Function OrientRectangle( const R : TRect) : TRect
12346: Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12347: Function PolygonBounds( const P : array of TPoint) : TRect
12348: Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12349: Function RGBValue( const Color : TColor) : TRGB
12350: Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12351: Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12352: Function PointAtDistance( AFrom, ATo : TPoint; Adist : Integer) : TPoint
12353: Function TeeCull( const P : TFourPoints) : Boolean;
12354: Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12355: TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12356: Procedure SmoothStretch( Src, Dst : TBitmap);
12357: Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12358: Function TeeDistance( const x, y : Double) : Double
12359: Function TeeLoadLibrary( const FileName : String) : HInst
12360: Procedure TeeFreeLibrary( hLibModule : HMODULE)
12361: Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12362: //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12363: Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12364:   SIRegister_ICanvasHyperlinks(CL);
12365:   SIRegister_ICanvasToolTips(CL);
12366:   Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12367: end;
12368:
12369: procedure SIRegister_ovcmisc(CL: TPPSPascalCompiler);
12370: begin
12371:   TOvcHdc', 'Integer
12372:   TOvcHWND', 'Cardinal
12373:   TOvcHdc', 'HDC
12374:   TOvcHWND', 'HWND
12375:   Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12376:   Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12377:   Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12378:   Function DefaultEpoch : Integer
12379:   Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12380:   Procedure FixRealPrim( P : PChar; DC : Char)
12381:   Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12382:   Function GetLeftButton : Byte
12383:   Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12384:   Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12385:   Function GetShiftFlags : Byte
12386:   Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont

```

```

12387: Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12388: Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet ) : string
12389: Function ovIsForegroundTask : Boolean
12390: Function ovTrimLeft( const S : string ) : string
12391: Function ovTrimRight( const S : string ) : string
12392: Function ovQuotedStr( const S : string ) : string
12393: Function ovWordCount( const S : string; const WordDelims : TCharSet ) : Integer
12394: Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12395: Function PtrDiff( const P1, P2 : PChar ) : Word
12396: Procedure PtrInc( var P, Delta : Word )
12397: Procedure PtrDec( var P, Delta : Word )
12398: Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer )
12399: Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY, SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer )
12400: Function ovMinI( X, Y : Integer ) : Integer
12401: Function ovMaxI( X, Y : Integer ) : Integer
12402: Function ovMinL( X, Y : LongInt ) : LongInt
12403: Function ovMaxL( X, Y : LongInt ) : LongInt
12404: Function GenerateComponentName( PF : TWInControl; const Root : string ) : string
12405: Function PartialCompare( const S1, S2 : string ) : Boolean
12406: Function PathEllipsis( const S : string; MaxWidth : Integer ) : string
12407: Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor ) : TBitmap
12408: Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas )
12409: Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap; TransparentColor : TColor )
12410: Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;TransparentColor : TColorRef)
12411: Function ovWidthOf( const R : TRect ) : Integer
12412: Function ovHeightOf( const R : TRect ) : Integer
12413: Procedure ovDebugOutput( const S : string )
12414: Function GetArrowWidth( Width, Height : Integer ) : Integer
12415: Procedure StripCharSeq( CharSeq : string; var Str : string )
12416: Procedure StripCharFromEnd( aChr : Char; var Str : string )
12417: Procedure StripCharFromFront( aChr : Char; var Str : string )
12418: Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12419: Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12420: Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12421: Function CreateEllipticRgn( p1, p2, p3, p4 : Integer ) : HRGN
12422: Function CreateEllipticRgnIndirect( const p1 : TRect ) : HRGN
12423: Function CreateFontIndirect( const p1 : TLogFont ) : HFONT
12424: Function CreateMetaFile( p1 : PChar ) : HDC
12425: Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12426: Function DrawText(hdc: HDC;lpString: PChar;nCount: Integer; var lpRect : TRect;uFormat:UINT):Integer
12427: Function DrawTextS(hdc: HDC;lpString:string;nCount: Integer; var lpRect: TRect;uFormat:UINT):Integer
12428: Function SetMapperFlags( DC : HDC; Flag : DWORD ) : DWORD
12429: Function SetGraphicsMode( hdc : HDC; iMode : Integer ) : Integer
12430: Function SetMapMode( DC : HDC; p2 : Integer ) : Integer
12431: Function SetMetaFileBitsEx( Size : UINT; const Data : PChar ) : HMETAFILE
12432: //Function SetPaletteEntries(Palette:HPalette;startIndex,NumEntries:UINT;var PaletteEntries):UINT
12433: Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF ) : COLORREF
12434: Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF ) : BOOL
12435: //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor ) : BOOL
12436: Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer ) : Integer
12437: Function StretchBlt(DestDC: HDC; X, Y, Width, Height: Int; SrcDC: HDC; XSrc, YSrc, SrcWidth, SrcHeight: Int; Rop:DWORD):BOOL
12438: Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer ) : BOOL
12439: Function StretchDIBits(dc : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth, SrcHeight:Int/Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD ) : Integer
12440: Function SetROP2( DC : HDC; p2 : Integer ) : Integer
12441: Function SetStretchBltMode( DC : HDC; StretchMode : Integer ) : Integer
12442: Function SetSystemPaletteUse( DC : HDC; p2 : UINT ) : UINT
12443: Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer ) : Integer
12444: Function SetTextColor( DC : HDC; Color : COLORREF ) : COLORREF
12445: Function SetTextAlign( DC : HDC; Flags : UINT ) : UINT
12446: Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer ) : Integer
12447: Function UpdateColors( DC : HDC ) : BOOL
12448: Function GetViewportExtEx( DC : HDC; var Size : TSize ) : BOOL
12449: Function GetViewportOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12450: Function GetWindowExtEx( DC : HDC; var Size : TSize ) : BOOL
12451: Function GetWindowOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12452: Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer ) : Integer
12453: Function InvertRgn( DC : HDC; p2 : HRGN ) : BOOL
12454: Function MaskBlt(DestDC: HDC; XDest, YDest, Width, Height: Integer; SrcDC : HDC; XScr, YScr : Integer; Mask : HBITMAP; xMask, yMask : Integer; Rop : DWORD ) : BOOL
12455: Function PlgBlt(DestDC: HDC; const PtsArray, SrcDC: HDC; XSrc, YSrc, Widt, Heigh: Int; Mask: HBITMAP; xMask, yMask: Int):BOOL;
12456: Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer ) : Integer
12457: Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer ) : Integer
12458: Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD ) : BOOL
12459: Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer ) : BOOL
12460: Function PlayMetaFile( DC : HDC; MF : HMETAFILE ) : BOOL
12461: Function PaintRgn( DC : HDC; RGN : HRGN ) : BOOL
12462: Function PtInRegion( RGN : HRGN; X, Y : Integer ) : BOOL
12463: Function PtVisible( DC : HDC; X, Y : Integer ) : BOOL
12464: Function RectInRegion( RGN : HRGN; const Rect : TRect ) : BOOL
12465: Function RectVisible( DC : HDC; const Rect : TRect ) : BOOL
12466: Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer ) : BOOL
12467: Function RestoreDC( DC : HDC; SavedDC : Integer ) : BOOL
12468: end;

```

```

12469:
12470: procedure SIRегистер_ovcfiler(CL: TPSPPascalCompiler);
12471: begin
12472:   SIRегистер_TOvcAbstractStore(CL);
12473:   //PEXPropInfo', '^TExPropInfo // will not work
12474: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12475:   SIRегистер_TOvcPropertyList(CL);
12476:   SIRегистер_TOvcDataFiler(CL);
12477:   Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12478:   Procedure UpdateStoredList( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12479:   Function CreateStoredItem( const CompName, PropName : string) : string
12480:   Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12481:   //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12482: end;
12483:
12484: procedure SIRегистер_ovccoco(CL: TPSPPascalCompiler);
12485: begin
12486:   'ovsetsize','LongInt'( 16);
12487:   'etSyntax','LongInt'( 0);
12488:   'etSemantic','LongInt'( 1);
12489:   'chCR','Char #13);
12490:   'chLF','Char #10);
12491:   'chLineSeparator','chCR);
12492:   SIRегистер_TCocoError(CL);
12493:   SIRегистер_TCommentItem(CL);
12494:   SIRегистер_TCommentList(CL);
12495:   SIRегистер_TSymbolPosition(CL);
12496:   TGenListType', '( glNever, glAlways, glOnError )
12497:   TovBitSet', 'set of Integer
12498:   //PStartTable', '^TStartTable // will not work
12499:   'TovCharSet', 'set of AnsiChar
12500:   TAFTERGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12501:   TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12502:   TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12503:   TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12504:   TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12505:   +'osition; const Data : string; ErrorType : integer)
12506:   TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12507:   TGetCH', 'Function ( pos : longint) : char
12508:   TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12509:   SIRегистер_TCocoRScanner(CL);
12510:   SIRегистер_TCocoRGrammar(CL);
12511:   '_EF','Char #0);
12512:   '_TAB','Char').SetString( #09);
12513:   '_CR','Char').SetString( #13);
12514:   '_LF','Char').SetString( #10);
12515:   '_EL','',').SetString( _CR);
12516:   '_EOF','Char').SetString( #26);
12517:   'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12518:   'minErrDist','LongInt'( 2);
12519:   Function ovPadL( S : string; ch : char; L : integer) : string
12520: end;
12521:
12522: TFormatSettings = record
12523:   CurrencyFormat: Byte;
12524:   NegCurrFormat: Byte;
12525:   ThousandSeparator: Char;
12526:   DecimalSeparator: Char;
12527:   CurrencyDecimals: Byte;
12528:   DateSeparator: Char;
12529:   TimeSeparator: Char;
12530:   ListSeparator: Char;
12531:   CurrencyString: string;
12532:   ShortDateFormat: string;
12533:   LongDateFormat: string;
12534:   TimeAMString: string;
12535:   TimePMString: string;
12536:   ShortTimeFormat: string;
12537:   LongTimeFormat: string;
12538:   ShortMonthNames: array[1..12] of string;
12539:   LongMonthNames: array[1..12] of string;
12540:   ShortDayNames: array[1..7] of string;
12541:   LongDayNames: array[1..7] of string;
12542:   TwoDigitYearCenturyWindow: Word;
12543: end;
12544:
12545: procedure SIRегистер_OvcFormatSettings(CL: TPSPPascalCompiler);
12546: begin
12547:   Function ovFormatSettings : TFormatSettings
12548: end;
12549:
12550: procedure SIRегистер_ovcstr(CL: TPSPPascalCompiler);
12551: begin
12552:   TOvc CharSet', 'set of Char
12553:   ovBTable', 'array[0..255] of Byte
12554:   //BTable = array[0..{$IFDEF UNICODE}{$ENDIF}{$IFDEF {$HUGE_UNICODE_BMTABLE}}$FFFF{$ELSE}{$ENDIF}{$ENDIF}{$ELSE}{$ENDIF}{$ENDIF} of Byte;
12555:   Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12556:   Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar

```

```

12557: Function BinaryWPChar( Dest : PChar; W : Word ) : PChar
12558: Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12559: Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12560: Function BMSearchC(var Buffer,BufLength:Cardinal; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12561: Function CharStrtPChar( Dest : PChar; C : Char; Len : Cardinal ) : PChar
12562: Function DatabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12563: Function HexBPChar( Dest : PChar; B : Byte) : PChar
12564: Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12565: Function HexPtrPChar( Dest : PChar; P : TObject ) : PChar
12566: Function HexWPChar( Dest : PChar; W : Word) : PChar
12567: Function LoCaseChar( C : Char ) : Char
12568: Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12569: Function StrChDeletePrim( P : PChar; Pos : Cardinal ) : PChar
12570: Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal ) : PChar
12571: Function StrChPos( P : PChar; C : Char; var Pos : Cardinal ) : Boolean
12572: Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12573: Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal ) : PChar
12574: Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal ) : PChar
12575: Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal ) : PChar
12576: Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal ) : PChar
12577: Function StrStPos( P, S : PChar; var Pos : Cardinal ) : Boolean
12578: Function StrToIntLongPChar( S : PChar; var I : LongInt ) : Boolean
12579: Procedure TrimAllSpacesPChar( P : PChar )
12580: Function TrimEmbeddedZeros( const S : string ) : string
12581: Procedure TrimEmbeddedZerosPChar( P : PChar )
12582: Function TrimTrailPrimPChar( S : PChar ) : PChar
12583: Function TrimTrailPChar( Dest, S : PChar ) : PChar
12584: Function TrimTrailingZeros( const S : string ) : string
12585: Procedure TrimTrailingZerosPChar( P : PChar )
12586: Function UpCaseChar( C : Char ) : Char
12587: Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet ) : Boolean
12588: Function ovc32StringIsCurrentCodePage( const S : WideString ) : Boolean;
12589: //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal ) : Boolean;
12590: end;
12591:
12592: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12593: begin
12594:   //PraiseFrame', '^TRaiseFrame // will not work
12595:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : __Poin'
12596:   +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12597:   Procedure SafeCloseHandle( var Handle : THandle )
12598:   Procedure ExchangeInteger( X1, X2 : Integer )
12599:   Procedure FillInteger( const Buffer, Size, Value : Integer )
12600:   Function LongMulDiv( Multi, Mult2, Div1 : Longint ) : Longint
12601:   Function afCompareMem( P1, P2 : TObject; Length : Integer ) : Boolean
12602:
12603:   FILENAME_ADVAPI32      = 'ADVAPI32.DLL';
12604:   function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12605:   function AbortSystemShutdown(lpMachineName: PKOLOChar): BOOL; stdcall;
12606:   function AccessCheckAndAuditAlarm(SubsystemName: PKOLOChar;
12607:     HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12608:     SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12609:     const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12610:     var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12611:   function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLOChar;
12612:     HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12613:     SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12614:     AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12615:     ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12616:     var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12617:   function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLOChar;
12618:     HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12619:     SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12620:     AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12621:     ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12622:     var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12623:   function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLOChar): BOOL; stdcall;
12624:   function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLOChar): BOOL; stdcall;
12625:   function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLOChar;
12626:     lpCommandLine: PKOLOChar; lpProcessAttributes: PSecurityAttributes;
12627:     lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12628:     dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLOChar;
12629:     const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12630:   function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12631:   function GetFileSecurity(lpFileName: PKOLOChar; RequestedInformation: SECURITY_INFORMATION;
12632:     pSecurityDescriptor: PSecurityDescriptor; nLength:DWORD; var lpnLengthNeeded: DWORD):BOOL; stdcall;
12633:   function GetUserName(lpBuffer: PKOLOChar; var nSize: DWORD): BOOL; stdcall;
12634:   function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLOChar;
12635:     dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12636:   function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLOChar;
12637:     dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12638:   function LookupAccountName(lpSystemName, lpAccountName: PKOLOChar;
12639:     Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLOChar;
12640:     var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12641:   function LookupAccountSid(lpSystemName: PKOLOChar; Sid: PSID;
12642:     Name: PKOLOChar; var cbName: DWORD; ReferencedDomainName: PKOLOChar;
12643:     var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12644:   function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLOChar;
12645:     lpDisplayName: PKOLOChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;

```

```

12646:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12647:         var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12648:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12649:         var lpLuid: TLargeInteger): BOOL; stdcall;
12650:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12651:         HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12652:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12653:         HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12654:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12655:         ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12656:         ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12657:         var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12658:         var GenerateOnClose: BOOL): BOOL; stdcall;
12659:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12660:         HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12661:         var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12662:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12663:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12664:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12665:         ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12666:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12667:         lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12668:         var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12669:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12670:         var phkResult: HKEY): Longint; stdcall;
12671:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12672:         var phkResult: HKEY): Longint; stdcall;
12673:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12674:         Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12675:         lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12676:         lpdwDisposition: PDWORD): Longint; stdcall;
12677:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12678:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12679:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12680:         var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12681:         lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12682:     function RegEnumKey(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar; cbName: DWORD): Longint; stdcall;
12683:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12684:         var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12685:         lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12686:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12687:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar; var phkResult: HKEY): Longint; stdcall;
12688:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12689:         ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12690:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12691:         lpcbClass: PDWORD; lpReserved: Pointer;
12692:         lpcbSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12693:         lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12694:         lpftLastWriteTime: PFileTime): Longint; stdcall;
12695:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12696:         NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotSize: DWORD): Longint; stdcall;
12697:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12698:         lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12699:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12700:         lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12701:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12702:         lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12703:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12704:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12705:         lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12706:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12707:         dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12708:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12709:         Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12710:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12711:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12712:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12713:         dwEventID: DWORD; lpUserId: Pointer; wNumStrings: Word;
12714:         dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12715:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12716:         pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12717:
12718:     Function wAddAtom( lpString : PKOLChar ) : ATOM
12719:     Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL ) : THandle
12720:         //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
12721:         lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD ) : BOOL
12722:         //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig ) : BOOL
12723:         Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
12724:             lpString2 : PKOLChar; cchCount2 : Integer ) : Integer
12725:         Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL ) : BOOL
12726:             //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
12727:                 TFnProgressRoutine; lpData : Pointer; pbcancel : PBool; dwCopyFlags : DWORD ) : BOOL
12728:         Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes ) : BOOL
12729:         Function wCreateDirectoryEx(lpTemplateDirectory,
12730:             lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribs):BOOL;
12731:         Function wCreateEvent(lpEventName:PSecurityAttribs:PSecurityAttrib:bManualReset,
12732:             bInitialState:BOOL;lpName:PKOLChar):THandle;
12733:         Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
12734:             PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD; hTemplateFile:THandle ) : THandle

```

```

12729: Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar ) : THandle
12730: Function wCreateHardLink( lpFileName,
lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12731: Function CreateMailslot( lpName : PKOLChar; MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes ):THandle;
12732: Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes ) : THandle
12733: //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
lpProcessInfo:TProcessInformation):BOOL
12734: Function wCreateSemaphore( lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
Longint; lpName : PKOLChar ) : THandle
12735: Function wCreateWaitableTimer( lpTimerAttrbs:PSecurityAttrbs;bManualReset:BOOL;lpTimerName:PKOLChar ):THandle;
12736: Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar ) : BOOL
12737: Function wDeleteFile( lpFileName : PKOLChar ) : BOOL
12738: Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL ) : BOOL
12739: //Function
wEnumCalendarInfo( lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE ):BOOL;
12740: //Function wEnumDateFormats( lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD ) : BOOL
12741: //Function
wEnumResourceNames( hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lpParam:Longint):BOOL;
12742: //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lpParam:Longint):BOOL;
12743: //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD ) : BOOL
12744: //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD ) : BOOL
12745: //Function wEnumTimeFormats( lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD ):BOOL;
12746: Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD ) : DWORD
12747: Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar )
12748: //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12749: Function wFindAtom( lpString : PKOLChar ) : ATOM
12750: Function wFindFirstChangeNotification( lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD ):THandle;
12751: Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData ) : THandle
12752: //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindIndexInfoLevels; lpFindFileData :
Pointer; fSearchOp : TFindIndexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD ) : BOOL
12753: Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData ) : BOOL
12754: Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar ) : HRSRC
12755: Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word ) : HRSRC
12756: Function wFoldString( dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer ):Integer;
12757: //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer ) : DWORD
12758: Function wFreeEnvironmentStrings( EnvBlock : PKOLChar ) : BOOL
12759: Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer ) : UINT
12760: Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD ) : BOOL
12761: Function wGetCommandLine : PKOLChar
12762: //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD ) : DWORD
12763: Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD ) : BOOL
12764: Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : WORD ) : WORD
12765: //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer ) : Integer
12766: Function wGetCurrentDirectory( nBufLength : DWORD; lpBuffer : PKOLChar ) : DWORD
12767: //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSysteMTime; lpFormat : PKOLChar;
lpDateStr : PKOLChar; cchDate : Integer ) : Integer
12768: //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD ):BOOL
12769: Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD ) : BOOL
12770: //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger ) : BOOL
12771: Function wGetDriveType( lpRootPathName : PKOLChar ) : UINT
12772: Function wGetEnvironmentStrings : PKOLChar
12773: Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD ) : DWORD;
12774: Function wGetFileAttributes( lpFileName : PKOLChar ) : DWORD
12775: //Function
wGetFileAttributesEx( lpFileName:PKOLChar,fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12776: Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12777: //Function wGetLocaleInfo(Locale:LCID; LCTYPE:LCTYPE;lpLCDData:PKOLChar;cchData:Integer): Integer
12778: Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar ) : DWORD
12779: Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD ) : DWORD
12780: Function wGetModuleHandle( lpModuleName : PKOLChar ) : HMODULE
12781: //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD ) : BOOL
12782: //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
lpNumberStr : PKOLChar; cchNumber : Integer ) : Integer
12783: Function wGetPrivateProfileInt( lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12784: Function
wGetPrivateProfileSection(lpAppName:PKOLChar;lpRtrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12785: Function wGetPrivateProfileSectionNames(lpszReturnBuffe:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12786: Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr : PKOLChar;
nSize:DWORD; lpFileName : PKOLChar ) : DWORD
12787: Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer ) : UINT
12788: Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD ) : DWORD
12789: Function wGetProfileString(lpAppName,lpKeyName,
lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12790: Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD ) : DWORD

```

```

12791: //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12792: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
12793: 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: Procedure wOutputDebugString( lpOutputString : PKOLChar )
12820: //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
12821: lpNumberOfEventsRead:DWORD):BOOL;
12822: Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD ) : DWORD
12823: //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12824: //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
12825: lpNumberOfCharsRead : DWORD; lpReserved : Pointer ) : BOOL
12826: //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
12827: lpNumOfEventsRead:DWORD):BOOL;
12828: //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
12829: : TCoord; var lpReadRegion : TSmallRect ) : BOOL
12830: //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
12831: DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD ) : BOOL
12832: Function wRemoveDirectory( lpPathName : PKOLChar ) : BOOL
12833: //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
12834: lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo ) : BOOL
12835: Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
12836: lpFilePart:PKOLChar):DWORD;
12837: Function wSetComputerName( lpComputerName : PKOLChar ) : BOOL
12838: Function wSetConsoleTitle( lpConsoleTitle : PKOLChar ) : BOOL
12839: Function wSetCurrentDirectory( lpPathName : PKOLChar ) : BOOL
12840: //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCCommConfig; dwSize : DWORD ) : BOOL
12841: Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar ) : BOOL
12842: Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD ) : BOOL
12843: //Function wSetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCDat : PKOLChar ) : BOOL
12844: Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar ) : BOOL
12845: //Function wUpdateResource(hUpdate:THandle,lpType,
12846: lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12847: Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD ) : DWORD
12848: Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD ) : BOOL
12849: //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
12850: DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer ) : BOOL
12851: //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
12852: var lpNumberOfEventsWritten : DWORD ) : BOOL
12853: //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
12854: TCoord; var lpWriteRegion : TSmallRect ) : BOOL
12855: //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
12856: DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12857: Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar ) : BOOL
12858: Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar ) : BOOL
12859: Function wWriteProfileSection( lpAppName, lpString : PKOLChar ) : BOOL
12860: Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar ) : BOOL
12861: Function wlstrcat( lpString1, lpString2 : PKOLChar ) : PKOLChar
12862: Function wlstrcmp( lpString1, lpString2 : PKOLChar ) : Integer
12863: Function wlstrcmpi( lpString1, lpString2 : PKOLChar ) : Integer
12864: Function wlstrcpy( lpString1, lpString2 : PKOLChar ) : PKOLChar
12865: Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer ) : PKOLChar
12866: Function wlstrlen( lpString : PKOLChar ) : Integer
12867: //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruct :
12868: PNetConnectInfoStruct ) : DWORD
12869: //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
12870: lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12871: //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
12872: lpUserName:PKOLChar; dwFlags : DWORD ) : DWORD
12873: Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar ) : DWORD
12874: Function wWNetCancelConnection( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL ) : DWORD
12875: Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL ) : DWORD
12876: //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct ) : DWORD
12877: //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct ) : DWORD

```

```

12859: //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12860: Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12861: Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
: PKOLChar; nNameBufSize : DWORD) : DWORD
12862: //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12863: Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12864: //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12865: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
lpBufferSize:DWORD):DWORD;
12866: Function wWNet GetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12867: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12868: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer ) : DWORD
12869: //Function wWNetUseConnection(hwndOwner:HWND;var
lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12870: Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12871: Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12872: Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12873: Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12874: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12875: //Func wGetPrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12876: //Func wWritePrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12877: Function wAddFontResource(FileName : PKOLChar) : Integer
12878: //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12879: Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12880: Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12881: //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12882: //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12883: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12884: Function wCreateFont( nHeight, nWidth, nEscapement, nOrientation, fnWeight : Integer; fdwItalic,
fdwUnderline, fdwStrikeOut, fdwCharSet, fdwOutputPrec, fdwClipPrecision, fdwQualy,
fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT
12885: Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12886: //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12887: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12888: Function wCreateMetaFile( p1 : PKOLChar) : HDC
12889: Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12890: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
pPort:PKOLChar;iIdx:Int;Out:PKOLChar;DevMod:PDeviceMode):Int;
12891: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFnFontEnumProc; p4 : LPARAM) : BOOL
12892: //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL;
12893: //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntermprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12894: //Function wEnumICMPprofiles( DC : HDC; ICMPProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12895: //Function wExtTextOut(DC:HDC;X,
Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12896: //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12897: //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12898: //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12899: //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12900: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12901: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12902: Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12903: Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12904: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12905: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:PGlyphMetrics; cbBuffer : DWORD;
lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12906: Function wGetICMPProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12907: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12908: Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12909: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12910: //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12911: //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12912: Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12913: Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12914: Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12915: //Function wGetTextMetrics( DC : HDC; var TM : TTTextMetric) : BOOL
12916: Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12917: Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12918: //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12919: //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12920: Function wSetICMPProfile( DC : HDC; Name : PKOLChar) : BOOL
12921: //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12922: Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12923: Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12924: Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12925: //Function wwglUseFontOutlines(p1:DC;p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12926: Function wAppendMenu( hMenu : HMENU; uFlags, uidNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12927: Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12928: //Function
wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12929: //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12930: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
dwFlags : DWORD; lParam : Pointer) : Longint
12931: Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12932: Function wCharLower( lpsz : PKOLChar) : PKOLChar

```

```

12933: Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12934: Function wCharNext( lpsz : PKOLChar ) : PKOLChar
12935: //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12936: Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar ) : PKOLChar
12937: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12938: Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12939: Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12940: Function wCharUpper( lpsz : PKOLChar ) : PKOLChar
12941: Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12942: Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer ) : Integer
12943: Function wCreateAcceleratorTable( var Accel, Count : Integer ) : HACCEL
12944: //Function wCreateDesktop(lpszDesktop,
12945: lpszDevice:PKOLChar,pDevMode:PDeviceMode,dwFlags:DWORD,dwDesiredAccess:DWORD;lpsa:PSecurityAttrs):HDESK
12946: //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
12947: HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND
12948: //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
12949: lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND
12950: Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
12951: hWndParent : HWND; hInstance : HINST; lParam : LPARAM ) : HWND
12952: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
12953: nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12954: //Function wCreateWindowStation(lpinsta:PKOLChar;dwReserv,
12955: dwDesiredAccess:DWORD;lpsa:PSecurityAttrs):HWINSTA;
12956: Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12957: Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12958: Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT;
12959: Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12960: //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
12961: : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12962: //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
12963: : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12964: Function wDispatchMessage( const lpMsg : TMsg ) : Longint
12965: Function wDlgsirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDLListBox,
nIDStaticPath:Integer;uFileType:UINT):Integer;
12966: Function wDlgsirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
nIDStaticPath:Int;uFileType:UINT):Int;
12967: Function wDlgsirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12968: Function wGetClassLong( hWnd : HWND; nIndex : Integer ) : DWORD
12969: Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer ) : Integer
12970: Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12971: Function wGetDlgItemText( hDlg : HWND;nIDDlgleItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12972: Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer ) : Integer
12973: Function wGetKeyboardLayoutName( pwszKLID : PKOLChar ) : BOOL
12974: //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo ) : BOOL
12975: Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12976: Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT ) : BOOL
12977: Function wGetProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12978: //Function wGetTabbedTextExtent( HDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
12979: lpnTabStopPositions ) : DWORD
12980: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
12981: lpnLengthNeed:DWORD)BOOL;
12982: Function wGetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
12983: Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT ) : UINT
12984: Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer ) : Integer
12985: Function wGetWindowTextLength( hWnd : HWND ) : Integer
12986: //Function wGrayString(hDC:HDC;hBrush:HBRUSH,lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
nHeight:Int):BOOL;
12987: Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12988: //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
12989: Function wIsCharAlpha( ch : KOLChar ) : BOOL
12990: Function wIsCharAlphaNumeric( ch : KOLChar ) : BOOL
12991: Function wIsCharLower( ch : KOLChar ) : BOOL
12992: Function wIsCharUpper( ch : KOLChar ) : BOOL
12993: Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar ) : HACCEL
12994: Function wLoadBitmap( hInst : HINST; lpBitmapName : PKOLChar ) : HBITMAP
12995: Function wLoadCursor( hInst : HINST; lpCursorName : PKOLChar ) : HCURSOR
12996: Function wLoadCursorFromFile( lpFileName : PKOLChar ) : HCURSOR
12997: Function wLoadIcon( hInst : HINST; lpIconName : PKOLChar ) : HICON
12998: Function wLoadImage(hInst:HMENUP;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THHandle
12999: Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT ) : HKL
13000: //Function wLoadMenuItemIndirect( lpMenuItemTemplate : Pointer ) : HMENU
13001: Function wLoadString(hInst:HMENUP;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
13002: Function wMapVirtualKey( uCode, uMapType : UINT ) : UINT
13003: Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL ) : UINT
13004: Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT ) : Integer
13005: Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
13006: //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams ) : BOOL

```

```

13007: Function wModifyMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
13008: //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR ) : BOOL
13009: //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD ) : BOOL
13010: //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
13011: Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : 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 wRealGetWindowClass( hWnd : HWND; pszType : PKOLChar; cchType : UINT ) : UINT
13018: // Function wRegisterClass( const lpWndClass : TWndClass ) : ATOM
13019: // Function wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM
13020: Function wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
13021: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13022: Function wRegisterWindowMessage( lpString : PKOLChar ) : UINT
13023: Function wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THandle
13024: Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13025: Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
13026: //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
lpResultCallBack : TFNSendAsyncProc; dwData : DWORD ) : BOOL
13027: Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
lpdwResult:DWORD) : LRESULT
13028: Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13029: Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
13030: Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar ) : BOOL
13031: //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
13032: Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle ) : BOOL
13033: // Function wSetUserObjectInformation(hObject:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
13034: Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint
13035: Function wSetWindowText( hWnd : HWND; lpString : PKOLChar ) : BOOL
13036: //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc ) : HHOOK
13037: //Function wSetWindowsHookEx(idHook:Integer,lpfn:TFNHookProc,hmod:HINST;dwThreadId:DWORD):HHOOK;
13038: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
13039: Function wTabbedTextOut(hdc:HDC;x,y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
lpnTabStopPositions,nTabOrigin:Int):Longint;
13040: Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg ) : Integer
13041: Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST ) : BOOL
13042: Function wVKeyScan( ch : KOLChar ) : SHORT
13043: Function wVKeyScanEx( ch : KOLChar; dwhkl : HKL ) : SHORT
13044: Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD ) : BOOL
13045: Function wwsprintf( Output : PKOLChar; Format : PKOLChar ) : Integer
13046: Function wwvssprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list ) : Integer
13047:
13048: //TestDrive!
13049: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL'
13050: 'PROC_CONVERTSINTOSTRINGSIDA','String').SetString( 'ConvertSintToStringSidA'
13051: Function GetDomainUserSids(const domainName:String;const userName:String; var foundDomain:String):String;
13052: Function GetlocalUserSidStr( const UserName : string ) : string
13053: Function getPid4user( const domain : string; const user : string; var pid : dword ) : boolean
13054: Function Impersonate2User( const domain : string; const user : string ) : boolean
13055: Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString ) : Boolean
13056: Function KillProcessbyname( const exename : string; var found : integer ) : integer
13057: Function getWinProcessList : TStringList
13058: function WaitTilClose(hWnd: Integer): Integer;
13059: function DoUserMsgs: Boolean;
13060: function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13061: procedure ShowMsg(hParent: Integer; const Mess, Title: String); //modal but NOT blockable
13062: procedure DeleteMsgForm(Handle: Integer);
13063: procedure DisableForms;
13064: function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13065: end;
13066:
13067: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13068: begin
13069: 'AfMaxSyncSlots','LongInt'( 64 );
13070: 'AfSynchronizeTimeout','LongInt'( 2000 );
13071: TafSyncSlotID', 'DWORD
13072: TafSyncStatistics', 'record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end';
13073: TafSafeSyncEvent', 'Procedure ( ID : TafSyncSlotID )
13074: TafSafeDirectSyncEvent', 'Procedure
13075: Function AfNewSyncSlot( const AEvent : TafSafeSyncEvent ) : TafSyncSlotID
13076: Function AfReleaseSyncSlot( const ID : TafSyncSlotID ) : Boolean
13077: Function AfEnableSyncSlot( const ID : TafSyncSlotID; Enable : Boolean ) : Boolean
13078: Function AfValidateSyncSlot( const ID : TafSyncSlotID ) : Boolean
13079: Function AfSyncEvent( const ID : TafSyncSlotID; Timeout : DWORD ) : Boolean
13080: Function AfDirectSyncEvent( Event : TafSafeDirectSyncEvent; Timeout : DWORD ) : Boolean
13081: Function AfIsSyncMethod : Boolean
13082: Function AfSyncWnd : HWnd
13083: Function AfSyncStatistics : TafSyncStatistics
13084: Procedure AfClearSyncStatistics
13085: end;
13086:
13087: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13088: begin
13089: 'fBinary','LongWord')($00000001);
13090: 'fParity','LongWord')($00000002);
13091: 'fOutxCtsFlow','LongWord').SetUInt($00000004);
13092: 'fOutxDsrFlow','LongWord')($00000008);

```

```

13093: 'fDtrControl','LongWord')($00000030);
13094: 'fDtrControlDisable','LongWord')($00000000);
13095: 'fDtrControlEnable','LongWord')($00000010);
13096: 'fDtrControlHandshake','LongWord')($00000020);
13097: 'fDsrsensitivity','LongWord')($00000040);
13098: 'fTxContinueOnKoff','LongWord')($00000080);
13099: 'fOutX','LongWord')($00000100);
13100: 'fInX','LongWord')($00000200);
13101: 'fErrorChar','LongWord')($00000400);
13102: 'fNull','LongWord')($00000800);
13103: 'fRtsControl','LongWord')($00003000);
13104: 'fRtsControlDisable','LongWord')($00000000);
13105: 'fRtsControlEnable','LongWord')($00001000);
13106: 'fRtsControlHandshake','LongWord')($00002000);
13107: 'fRtsControlToggle','LongWord')($00003000);
13108: 'fAbortOnError','LongWord')($00004000);
13109: 'fDummy2','LongWord')($FFFF8000);
13110: TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13111: FindClass('TObject'), 'EAfComPortCoreError
13112: FindClass('TObject'), 'TAfComPortCore
13113: TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Event'
13114: +'tKind : TAfCoreEvent; Data : DWORD)
13115: SIRegister_TAfComPortCoreThread(CL);
13116: SIRegister_TAfComPortEventThread(CL);
13117: SIRegister_TAfComPortWriteThread(CL);
13118: SIRegister_TAfComPortCore(CL);
13119: Function FormatDeviceName( PortNumber : Integer ) : string
13120: end;
13121:
13122: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13123: begin
13124:   TAPIOFileStreamEvent', 'Function ( const fileName : String; mode: Word ) : TStream
13125:   TAPIOFileExistsEvent', 'Function ( const fileName : String ) : Boolean
13126:   SIRegister_TApplicationFileIO(CL);
13127:   TDataFileCapability', '( dfcRead, dfcWrite )
13128:   TDataFileCapabilities', 'set of TDataFileCapability
13129:   SIRegister_TDatafile(CL);
13130:   //TDataFileClass', 'class of TDataFile
13131:   Function ApplicationFileIODefined : Boolean
13132:   Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13133:   Function FilestreamExists(const fileName: String) : Boolean
13134:   //Procedure Register
13135: end;
13136:
13137: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13138: begin
13139:   TALFBXFieldType', '( uftUnknown, uftNumeric, uftChar, uftVarchar'
13140:   +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecision'
13141:   +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13142:   TALFBXScale', 'Integer
13143:   FindClass('TObject'), 'EALFBXConvertError
13144:   SIRegister_EALFBXError(CL);
13145:   SIRegister_EALFBXException(CL);
13146:   FindClass('TObject'), 'EALFBXGFixError
13147:   FindClass('TObject'), 'EALFBXDSQLError
13148:   FindClass('TObject'), 'EALFBXDynError
13149:   FindClass('TObject'), 'EALFBXBakError
13150:   FindClass('TObject'), 'EALFBXGSecError
13151:   FindClass('TObject'), 'EALFBXLicenseError
13152:   FindClass('TObject'), 'EALFBXStatError
13153:   //EALFBXExceptionClass', 'class of EALFBXError
13154:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13155:   +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13156:   +'csEUUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13157:   +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252, '
13158:   +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13159:   +'SDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13160:   +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13161:   +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13162:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13163:   +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13164:   +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13165:   +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13166:   TALFBXTransParams', 'set of TALFBXTransParam
13167: Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString ) : TALFBXCharacterSet
13168: Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13169: Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13170: 'CALFBXMaxParamLength', 'LongInt'( 125 );
13171: TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13172: TALFBXParamsFlags', 'set of TALFBXParamsFlag
13173: //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13174: //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13175: TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete, '
13176:   +'stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommis'
13177:   +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13178: SIRegister_TALFBXSQLDA(CL);
13179: //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13180: SIRegister_TALFBXPoolStream(CL);
13181: //PALFBXBlobData', '^TALFBXBlobData // will not work

```

```

13182: TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13183: //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13184: //TALFBXArrayDesc', 'TISCArrayDesc
13185: //TALFBXBlobDesc', 'TISCBlobDesc
13186: //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13187: //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13188: SIRегистер_TALFBXSQLResult(CL);
13189: //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13190: SIRегистер_TALFBXSQLParams(CL);
13191: //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13192: TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13193: +'atementType : TALFBXStatementType; end
13194: FindClass('TObject'), TALFBXLibrary
13195: //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13196: TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary )
13197: //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13198: +'Excep : EALFBXExceptionClass)
13199: SIRегистер_TALFBXLibrary(CL);
13200: 'calfBXDateOffset', 'LongInt'( 15018 );
13201: 'calfBXTIMECoef', 'LongInt'( 864000000 );
13202: //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double );
13203: //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp );
13204: //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp ) : Double;
13205: Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word )
13206: Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13207: //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp );
13208: //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp );
13209: //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp );
13210: Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer ) : Integer
13211: Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord ) : Cardinal
13212: TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13213: TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13214: Function ALFBXSQLQuote( const name : AnsiString ) : AnsiString
13215: Function ALFBXSQLUnQuote( const name : AnsiString ) : AnsiString
13216: end;
13217:
13218: procedure SIRегистер_ALFBXClient(CL: TPSPPascalCompiler);
13219: begin
13220:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13221:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13222:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13223: +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13224: +'teger; First : Integer; CacheThreshold : Integer; end
13225: TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13226: TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13227: TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13228: TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13229: +'_writes : int64; page_fetches : int64; page_marks : int64; end
13230: SIRегистер_TALFBXClient(CL);
13231: SIRегистер_TALFBXConnectionStatementPoolBinTreeNode(CL);
13232: SIRегистер_TALFBXConnectionStatementPoolBinTree(CL);
13233: SIRегистер_TALFBXConnectionWithStmtPoolContainer(CL);
13234: SIRегистер_TALFBXConnectionWithoutStmtPoolContainer(CL);
13235: SIRегистер_TALFBXReadTransactionPoolContainer(CL);
13236: SIRегистер_TALFBXReadStatementPoolContainer(CL);
13237: SIRегистер_TALFBXStringKeyPoolBinTreeNode(CL);
13238: SIRегистер_TALFBXConnectionPoolClient(CL);
13239: SIRегистер_TALFBXEventThread(CL);
13240: Function AlMySqlClientSlashedStr( const Str : AnsiString ) : AnsiString
13241: end;
13242:
13243: procedure SIRегистер_ovcBidi(CL: TPSPPascalCompiler);
13244: begin
13245: _OSVERSIONINFOA = record
13246:   dwOSVersionInfoSize: DWORD;
13247:   dwMajorVersion: DWORD;
13248:   dwMinorVersion: DWORD;
13249:   dwBuildNumber: DWORD;
13250:   dwPlatformId: DWORD;
13251:   szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13252: end;
13253: TOSversionInfoA', '_OSVERSIONINFOA
13254: TOSversionInfo', 'TOSVersionInfoA
13255: 'WS_EX_RIGHT', 'LongWord')($00001000);
13256: 'WS_EX_LEFT', 'LongWord')($00000000);
13257: 'WS_EX_RTLREADING', 'LongWord')($00002000);
13258: 'WS_EX_LTRREADING', 'LongWord')($00000000);
13259: 'WS_EX_LEFTSCROLLBAR', 'LongWord')($00004000);
13260: 'WS_EX_RIGHTSCROLLBAR', 'LongWord')($00000000);
13261: Function SetProcessDefaultLayout( dwDefaultLayout : DWORD ) : BOOL
13262: 'LAYOUT_RTL', 'LongWord')($00000001);
13263: 'LAYOUT_BTT', 'LongWord')($00000002);
13264: 'LAYOUT_VBH', 'LongWord')($00000004);
13265: 'LAYOUT_BITMAPORIENTATIONPRESERVED', 'LongWord')($00000008);
13266: 'NOMIRRORBITMAP', 'LongWord')($00000000);
13267: Function SetLayout( dc : HDC; dwLayout : DWORD ) : DWORD
13268: Function GetLayout( dc : hdc ) : DWORD
13269: Function IsBidi : Boolean
13270: Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo ) : BOOL

```

```

13271: Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13272: Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13273: Function GetPriorityClass( hProcess : THandle) : DWORD
13274: Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13275: Function CloseClipboard : BOOL
13276: Function GetClipboardSequenceNumber : DWORD
13277: Function GetClipboardOwner : HWND
13278: Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13279: Function GetClipboardViewer : HWND
13280: Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13281: Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13282: Function GetClipboardData( uFormat : UINT) : THandle
13283: Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13284: Function CountClipboardFormats : Integer
13285: Function EnumClipboardFormats( format : UINT) : UINT
13286: Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13287: Function EmptyClipboard : BOOL
13288: Function IsClipboardFormatAvailable( format : UINT) : BOOL
13289: Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13290: Function GetOpenClipboardWindow : HWND
13291: Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13292: Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13293: Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13294: Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13295: Function SetDlgItemText( hDlg : HWND; nIDButton : Integer; lpString : PChar) : BOOL
13296: Function CheckDlgButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13297: Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13298: Function IsDlgItemChecked( hDlg : HWND; nIDButton : Integer) : UINT
13299: Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13300: end;
13301:
13302: procedure SIRегистre_DXPUtils(CL: TPSPascalCompiler);
13303: begin
13304:   Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13305:   Function GetTemporaryFilesPath : String
13306:   Function GetTemporaryFileName : String
13307:   Function FindFileInPaths( const fileName, paths : String) : String
13308:   Function PathsToString( const paths : TStrings) : String
13309:   Procedure StringToPaths( const pathsString : String; paths : TStrings)
13310: //Function MacroExpandPath( const aPath : String) : String
13311: end;
13312:
13313: procedure SIRегистre_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13314: begin
13315:   SIRегистre_TALMultiPartBaseContent(CL);
13316:   SIRегистre_TALMultiPartBaseContents(CL);
13317:   SIRегистre_TALMultiPartBaseStream(CL);
13318:   SIRегистre_TALMultiPartBaseEncoder(CL);
13319:   SIRегистre_TALMultiPartBaseDecoder(CL);
13320:   Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13321:   Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13322:   Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13323: end;
13324:
13325: procedure SIRегистre_SmallUtils(CL: TPSPascalCompiler);
13326: begin
13327:   TdriveSize', 'record FreeS : Int64; TotalsS : Int64; end
13328:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In
13329:   +teger; WinMinorVersion : Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13330:   Function aAllocPadedMem( Size : Cardinal) : TObject
13331:   Procedure aFreePadedMem( var P : TObject);
13332:   Procedure aFreePadedMem( var P : PChar);
13333:   Function aCheckPadedMem( P : Pointer) : Byte
13334:   Function aGetPadMemSize( P : Pointer) : Cardinal
13335:   Function aAllocMem( Size : Cardinal) : Pointer
13336:   Function aStrLen( const Str : PChar) : Cardinal
13337:   Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13338:   Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13339:   Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13340:   Function aStrEnd( const Str : PChar) : PChar
13341:   Function aStrScan( const Str : PChar; aChr : Char) : PChar
13342:   Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13343:   Function aPCharLength( const Str : PChar) : Cardinal
13344:   Function aPCharUpper( Str : PChar) : PChar
13345:   Function aPCharLower( Str : PChar) : PChar
13346:   Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13347:   Function aLastDelimiter( const Delimiters, S : String) : Integer
13348:   Function aCopyTail( const S : String; Len : Integer) : String
13349:   Function aInt2Thos( I : Int64) : String
13350:   Function aUpperCase( const S : String) : String
13351:   Function aLowerCase( const S : string) : String
13352:   Function aCompareText( const S1, S2 : string) : Integer
13353:   Function aSameText( const S1, S2 : string) : Boolean
13354:   Function aInt2Str( Value : Int64) : String
13355:   Function aStr2Int( const Value : String) : Int64
13356:   Function aStr2IntDef( const S : string; Default : Int64) : Int64
13357:   Function aGetFileExt( const FileName : String) : String
13358:   Function aGetFilePath( const FileName : String) : String
13359:   Function aGetFileName( const FileName : String) : String

```

```

13360: Function aChangeExt( const FileName, Extension : String ) : String
13361: Function aAdjustLineBreaks( const S : string ) : string
13362: Function aGetWindowStr( WinHandle : HWND ) : String
13363: Function aDiskSpace( Drive : String ) : TdriveSize
13364: Function aFileExists( FileName : String ) : Boolean
13365: Function aFileSize( FileName : String ) : Int64
13366: Function aDirectoryExists( const Name : string ) : Boolean
13367: Function aSysErrorMessage( ErrorCode : Integer ) : string
13368: Function aShortPathName( const LongName : string ) : string
13369: Function aGetWindowVer : TWinVerRec
13370: procedure InitDriveSpacePtr;
13371: end;
13372:
13373: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13374: begin
13375:   aZero', 'LongInt'( 0 );
13376:   'makeappDEF', 'LongInt'( - 1 );
13377:   'CS_VREDRAW', 'LongInt'( DWORD ( 1 ) );
13378:   'CS_HREDRAW', 'LongInt'( DWORD ( 2 ) );
13379:   'CS_KEYCVTWINDOW', 'LongInt'( 4 );
13380:   'CS_DBLCLKS', 'LongInt'( 8 );
13381:   'CS_OWNDC', 'LongWord')( $20 );
13382:   'CS_CLASSDC', 'LongWord')( $40 );
13383:   'CS_PARENTDC', 'LongWord')( $80 );
13384:   'CS_NOKEYCWT', 'LongWord')( $100 );
13385:   'CS_NOCLOSE', 'LongWord')( $200 );
13386:   'CS_SAVEBITS', 'LongWord')( $800 );
13387:   'CS_BYTEALIGNCLIENT', 'LongWord')( $1000 );
13388:   'CS_BYTEALIGNWINDOW', 'LongWord')( $2000 );
13389:   'CS_GLOBALCLASS', 'LongWord')( $4000 );
13390:   'CS_IME', 'LongWord')( $10000 );
13391:   'CS_DROPSHADOW', 'LongWord')( $20000 );
13392:   //TPanelFunc', 'TPanelFunc // will not work
13393:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13394:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13395:   TFontLooks', 'set of TFontLook
13396:   TMessagefunc', 'function(hWnd,iMsg,wParam:lParam:Integer):Integer
13397:   Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13398:   Function SetWinClassO( const ClassName : String; pMessFunc : Tobject; wcStyle : Integer): Word
13399:   Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer ) : Word
13400:   Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13401:   Procedure RunMsgLoop( Show : Boolean )
13402:   Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13403:   Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
ID_Number:Cardinal;hFont:Int):Int;
13404:   Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13405:   Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13406:   Function MakePanel(Left,Top,Width,Height,
hParent:Int;hWndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13407:   Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13408:   Function id4menu( a, b : Byte; c : Byte; d : Byte ) : Cardinal
13409:   Procedure DoInitMakeApp //set first to init formclasscontrol!
13410: end;
13411:
13412: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13413: begin
13414:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13415:   +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13416:   TScreenSaverOptions', 'set of TScreenSaverOption
13417:   'cDefaultScreenSaverOptions', 'LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13418:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND )
13419:   SIRegister_TScreensaver(CL);
13420:   //Procedure Register
13421:   Procedure SetScreenSaverPassword
13422: end;
13423:
13424: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13425: begin
13426:   FindClass('TOBJECT'), 'TXCollection
13427:   SIRegister_EFilerException(CL);
13428:   SIRegister_TXCollectionItem(CL);
13429:   //TXCollectionItemClass', 'class of TXCollectionItem
13430:   SIRegister_TXCollection(CL);
13431:   Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent )
13432:   Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent )
13433:   Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass )
13434:   Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass )
13435:   Function FindXCollectionItemClass( const className : String ) : TXCollectionItemClass
13436:   Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass ) : TList
13437: end;
13438:
13439: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13440: begin
13441:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond, mtcmArbitrary);
13442:   Procedure xglMapTexCoordToNull
13443:   Procedure xglMapTexCoordToMain
13444:   Procedure xglMapTexCoordToSecond
13445:   Procedure xglMapTexCoordToDual

```

```

13446: Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13447: Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13448: Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13449: Procedure xglBeginUpdate
13450: Procedure xglEndUpdate
13451: Procedure xglPushState
13452: Procedure xglPopState
13453: Procedure xglForbidSecondTextureUnit
13454: Procedure xglAllowSecondTextureUnit
13455: Function xglGetBitWiseMapping : Cardinal
13456: end;
13457:
13458: procedure SIRegister_VectorLists(CL: TPSPPascalCompiler);
13459: begin
13460:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13461:   TBaseListOptions', 'set of TBaseListOption
13462:   SIRegister_TBaseList(CL);
13463:   SIRegister_TBaseVectorList(CL);
13464:   SIRegister_TAffineVectorList(CL);
13465:   SIRegister_TVectorList(CL);
13466:   SIRegister_TTexPointList(CL);
13467:   SIRegister_TXIntegerList(CL);
13468:   //PSingleArrayList', '^TSingleArrayList // will not work
13469:   SIRegister_TSingleList(CL);
13470:   SIRegister_TByteList(CL);
13471:   SIRegister_TQuaternionList(CL);
13472:   Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList );
13473:   Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList );
13474:   Procedure FastQuickSortLists(startIndex,
endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13475: end;
13476:
13477: procedure SIRegister_MeshUtils(CL: TPSPPascalCompiler);
13478: begin
13479:   Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList );
13480:   Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList );
13481:   Procedure ConvertStripToList2(const strip:TAffineVectorList;const
indices:TIntegerList;list:TAffineVectorList);
13482:   Procedure ConvertIndexedListToList( const data:TAffineVectlist;const
indices:TIntegerList;list:TAffineVectorList);
13483:   Function BuildVectorCountOptimizedIndices( const vertices:TAffineVectorList; const
normals:TAffineVectorList; const texCoords : TAffineVectorList ) : TIntegerList
13484:   Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList );
13485:   Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList );
13486:   Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList )
13487:   Function RemapIndicesToIndicesMap( remapIndices : TIntegerList ) : TIntegerList
13488:   Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList )
13489:   Procedure RemapIndices( indices, indicesMap : TIntegerList )
13490:   Procedure UnifyTrianglesWinding( indices : TIntegerList )
13491:   Procedure InvertTrianglesWinding( indices : TIntegerList )
13492:   Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList ) : TAffineVectorList
13493:   Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
edgesTriangles : TIntegerList ) : TIntegerList
13494:   Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single )
13495:   Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean);
TPersistentObjectList;
13496:   Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer )
13497:   Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent )
13498: end;
13499:
13500: procedure SIRegister_JclSysUtils(CL: TPSPPascalCompiler);
13501: begin
13502:   Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte )
13503:   Procedure FreeMemAndNil( var P : TObject )
13504:   Function PCharOrNil( const S : string ) : PChar
13505:   SIRegister_TJclReferenceMemoryStream(CL);
13506:   FindClass('TOBJECT'), EJclVMTError
13507:   {Function GetVirtualMethodCount( AClass : TClass ) : Integer
13508:   Function GetVirtualMethod( AClass : TClass; const Index : Integer ) : Pointer
13509:   Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer )
13510:   PDynamicIndexList', '^TDynamicIndexList // will not work
13511:   PDynamicAddressList', '^TDynamicAddressList // will not work
13512:   Function GetDynamicMethodCount( AClass : TClass ) : Integer
13513:   Function GetDynamicIndexList( AClass : TClass ) : PDynamicIndexList
13514:   Function GetDynamicAddressList( AClass : TClass ) : PDynamicAddressList
13515:   Function HasDynamicMethod( AClass : TClass; Index : Integer ) : Boolean
13516:   Function GetDynamicMethod( AClass : TClass; Index : Integer ) : Pointer
13517:   Function GetInitTable( AClass : TClass ) : PTypeInfo
13518:   PFieldEntry', '^TFieldEntry // will not work
13519:   TFieldEntry', 'record Offset : Integer; IDX : Word; Name : ShortString; end
13520:   Function JIsClass( Address : Pointer ) : Boolean
13521:   Function JIsObject( Address : Pointer ) : Boolean
13522:   Function GetImplementorOfInterface( const I : IInterface ) : TObject
13523:   TdigitCount', 'Integer
13524:   SIRegister_TJclNumericFormat(CL);
13525:   Function JIntToStrZeroPad( Value, Count : Integer ) : AnsiString
13526:   TTextHandler', 'Procedure ( const Text : string )
13527: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
```

```

13528: Function JExecute(const
13529:   CommandLine:string;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Card;
13530: Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13531: Function ReadKey: Char //to and from the DOS console !
13531:   TModuleHandle, 'HINST
13532:   //TModuleHandle', 'Pointer
13533:   'INVALID_MODULEHANDLE_VALUE', 'LongInt'( TModuleHandle ( 0 ) );
13534: Function LoadModule( var Module : TModuleHandle; FileName : string ) : Boolean
13535: Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal ) : Boolean
13536: Procedure UnloadModule( var Module : TModuleHandle )
13537: Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string ) : Pointer
13538: Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean ) : Pointer
13539: Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13540: Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13541:   FindClass('TOBJECT'),'EJclConversionError
13542: Function JStrToBoolean( const S : string ) : Boolean
13543: Function JBooleanToStr( B : Boolean ) : string
13544: Function JIntToBool( I : Integer ) : Boolean
13545: Function JBoolToInt( B : Boolean ) : Integer
13546: 'ListSeparator','String '
13547: 'ListSeparator1','String :
13548: Procedure ListAddItems( var List : string; const Separator, Items : string )
13549: Procedure ListIncludeItems( var List : string; const Separator, Items : string )
13550: Procedure ListRemoveItems( var List : string; const Separator, Items : string )
13551: Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer )
13552: Function ListItemCount( const List, Separator : string ) : Integer
13553: Function ListItem( const List, Separator : string; const Index : Integer ) : string
13554: Procedure ListItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13555: Function ListItemIndex( const List, Separator, Item : string ) : Integer
13556: Function SystemToObjectInstance : LongWord
13557: Function IsCompiledWithPackages : Boolean
13558: Function JJclGUIDToString( const GUID : TGUID ) : string
13559: Function JJclStringToGUID( const S : string ) : TGUID
13560: SIRegister_TJclIntfCriticalSection(CL);
13561: SIRegister_TJclSimpleLog(CL);
13562: Procedure InitSimpleLog( const ALogFileFileName : string )
13563: end;
13564:
13565: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13566: begin
13567:   FindClass('TOBJECT'),'EJclBorRADException
13568:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )'
13569:   TJclBorRADToolEdition', '( deOPEN, dePRO, desVR )'
13570:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )'
13571:   TJclBorRADToolPath', 'string
13572:   'SupportedDelphiVersions','LongInt'( 5 or 6 or 7 or 8 or 9 or 10 or 11 );
13573:   'SupportedBCBVersions','LongInt'( 5 or 6 or 10 or 11 );
13574:   'SupportedBDSVersions','LongInt'( 1 or 2 or 3 or 4 or 5 );
13575:   BorRADToolRepositoryPagesSection,'String 'Repository Pages
13576:   BorRADToolRepositoryDialogsPage,'String 'Dialogs
13577:   BorRADToolRepositoryFormsPage,'String 'Forms
13578:   BorRADToolRepositoryProjectsPage,'String 'Projects
13579:   BorRADToolRepositoryDataModulesPage,'String 'Data Modules
13580:   BorRADToolRepositoryObjectType,'String 'Type
13581:   BorRADToolRepositoryFormTemplate,'String 'FormTemplate
13582:   BorRADToolRepositoryProjectTemplate,'String 'ProjectTemplate
13583:   BorRADToolRepositoryObjectName,'String 'Name
13584:   BorRADToolRepositoryObjectPage,'String 'Page
13585:   BorRADToolRepositoryObjectIcon,'String 'Icon
13586:   BorRADToolRepositoryObjectDescr,'String 'Description
13587:   BorRADToolRepositoryObjectAuthor,'String 'Author
13588:   BorRADToolRepositoryObjectAncestor,'String 'Ancestor
13589:   BorRADToolRepositoryObjectDesigner,'String 'Designer
13590:   BorRADToolRepositoryDesignerDfm,'String 'dfm
13591:   BorRADToolRepositoryDesignerXfm,'String 'xmf
13592:   BorRADToolRepositoryObjectNewForm,'String 'DefaultNewForm
13593:   BorRADToolRepositoryObjectMainForm,'String 'DefaultMainForm
13594:   SourceExtensionDelphiPackage,'String '.dpk
13595:   SourceExtensionBCBPackage,'String '.bpk
13596:   SourceExtensionDelphiProject,'String '.dpr
13597:   SourceExtensionBCBProject,'String '.bpr
13598:   SourceExtensionBDSProject,'String '.bdsproj
13599:   SourceExtensionDProject,'String '.dproj
13600:   BinaryExtensionPackage,'String '.bpl
13601:   BinaryExtensionLibrary,'String '.dll
13602:   BinaryExtensionExecutable,'String '.exe
13603:   CompilerExtensionDCP,'String '.dcp
13604:   CompilerExtensionBPI,'String '.bpi
13605:   CompilerExtensionLIB,'String '.lib
13606:   CompilerExtensionTDS,'String '.tds
13607:   CompilerExtensionMAP,'String '.map
13608:   CompilerExtensionDRC,'String '.drc
13609:   CompilerExtensionDEF,'String '.def
13610:   SourceExtensionCPP,'String '.cpp
13611:   SourceExtensionH,'String '.h
13612:   SourceExtensionPAS,'String '.pas
13613:   SourceExtensionDFM,'String '.dfm
13614:   SourceExtensionXFM,'String '.xfm
13615:   SourceDescriptionPAS,'String 'Pascal source file

```

```

13616: SourceDescriptionCPP', 'String 'C++ source file
13617: DesignerVCL', 'String 'VCL
13618: DesignerCLX', 'String 'CLX
13619: ProjectTypePackage', 'String 'package
13620: ProjectTypeLibrary', 'String 'library
13621: ProjectTypeProgram', 'String 'program
13622: Personality32Bit', 'String '32 bit
13623: Personality64Bit', 'String '64 bit
13624: PersonalityDelphi', 'String 'Delphi
13625: PersonalityDelphiDotNet', 'String 'Delphi.net
13626: PersonalityBCB', 'String 'C++Builder
13627: PersonalityCSB', 'String 'C#Builder
13628: PersonalityVB', 'String 'Visual Basic
13629: PersonalityDesign', 'String 'Design
13630: PersonalityUnknown', 'String 'Unknown personality
13631: PersonalityBDS', 'String 'Borland Developer Studio
13632: DOFDirectoriesSection', 'String 'Directories
13633: DOFUnitOutputDirKey', 'String 'UnitOutputDir
13634: DOFSearchPathName', 'String 'SearchPath
13635: DOFConditionals', 'String 'Conditionals
13636: DOFLinkerSection', 'String 'Linker
13637: DOFPackagesKey', 'String 'Packages
13638: DOFCompilerSection', 'String 'Compiler
13639: DOFPackageNoLinkKey', 'String 'PackageNoLink
13640: DOFAdditionalSection', 'String 'Additional
13641: DOFOptionsKey', 'String 'Options
13642: TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13643: '+pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13644: '+bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13645: TJclBorPersonalities', 'set of TJclBorPersonality
13646: TJclBorDesigner', '( bdVCL, bdCLX )
13647: TJclBorDesigners', 'set of TJclBorDesigner
13648: TJclBorPlatform', '( bp32bit, bp64bit )
13649: FindClass('TOBJECT'), 'TJclBorRADToolInstallation
13650: SIRegister_TJclBorRADToolInstallationObject(CL);
13651: SIRegister_TJclBorLandOpenHelp(CL);
13652: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13653: TJclHelp2Objects', 'set of TJclHelp2Object
13654: SIRegister_TJclHelp2Manager(CL);
13655: SIRegister_TJclBorRADToolIDETool(CL);
13656: SIRegister_TJclBorRADToolIDEPackages(CL);
13657: SIRegister_IJclCommandLineTool(CL);
13658: FindClass('TOBJECT'), 'EJclCommandLineToolError
13659: SIRegister_TJclCommandLineTool(CL);
13660: SIRegister_TJclBorLandCommandLineTool(CL);
13661: SIRegister_TJclBCC32(CL);
13662: SIRegister_TJclDCC32(CL);
13663: TJclDCC', 'TJclDCC32
13664: SIRegister_TJclBpr2Mak(CL);
13665: SIRegister_TJclBorLandMake(CL);
13666: SIRegister_TJclBorRADToolPalette(CL);
13667: SIRegister_TJclBorRADToolRepository(CL);
13668: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
13669: TCommandLineTools', 'set of TCommandLineTool
13670: //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13671: SIRegister_TJclBorRADToolInstallation(CL);
13672: SIRegister_TJclBCBInstallation(CL);
13673: SIRegister_TJclDelphiInstallation(CL);
13674: SIRegister_TJclDCCIL(CL);
13675: SIRegister_TJclBDSInstallation(CL);
13676: TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation ) : Boolean
13677: SIRegister_TJclBorRADToolInstallations(CL);
13678: Function BPLFileName( const BPLPath, PackageFileName : string ) : string
13679: Function BinaryFileName( const OutputPath, ProjectFileName : string ) : string
13680: Function IsDelphiPackage( const FileName : string ) : Boolean
13681: Function IsDelphiProject( const FileName : string ) : Boolean
13682: Function IsBCBPackage( const FileName : string ) : Boolean
13683: Function IsBCBProject( const FileName : string ) : Boolean
13684: Procedure GetDPRFileInfo( const DPRFileName:string;out BinaryExtensio:string;const LibSuffix:PString );
13685: Procedure GetBPRFileInfo( const BPRFileName:string;out BinaryFileName:string;const Descript:PString );
13686: Procedure GetDPKFileInfo( const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
Descript:PString;
13687: Procedure GetBPKFileInfo( const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
Descript:PString
13688: function SamePath(const Path1, Path2: string): Boolean;
13689: end;
13690:
13691: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13692: begin
13693:   'ERROR_NO_MORE_FILES', LongInt( 18 );
13694:   //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13695:   //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64 ) : Integer
13696:   //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13697:   'LPathSeparator','String '/'
13698:   'LDirDelimiter','String '
13699:   'LDirSeparator','String '
13700:   'JXPathDevicePrefix','String '\.\.
13701:   'JXPathSeparator','String \
13702:   'JXDirDelimiter','String \

```

```

13703: 'JXDirSeparator','String ';
13704: 'JXPathUncPrefix','String '\\
13705: 'faNormalFile','LongWord')($00000080);
13706: //faUnixSpecific,' faSymLink';
13707: JXTCompactPath', '( cpCenter, cpEnd )
13708: _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13709: +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime : '
13710: +' TFileTime; nfileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13711: TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13712: WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13713:
13714: Function jxPathAddSeparator( const Path : string ) : string
13715: Function jxPathAddExtension( const Path, Extension : string ) : string
13716: Function jxPathAppend( const Path, Append : string ) : string
13717: Function jxPathBuildRoot( const Drive : Byte ) : string
13718: Function jxPathCanonicalize( const Path : string ) : string
13719: Function jxPathCommonPrefix( const Path1, Path2 : string ) : Integer
13720: Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13721: Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string )
13722: Function jxPathExtractFileDirName( const S : string ) : string
13723: Function jxPathExtractFileNameNoExt( const Path : string ) : string
13724: Function jxPathExtractPathDepth( const Path : string; Depth : Integer ) : string
13725: Function jxPathGetDepth( const Path : string ) : Integer
13726: Function jxPathGetLongName( const Path : string ) : string
13727: Function jxPathGetShortName( const Path : string ) : string
13728: Function jxPathGetLongName( const Path : string ) : string
13729: Function jxPathGetShortName( const Path : string ) : string
13730: Function jxPathGetRelativePath( Origin, Destination : string ) : string
13731: Function jxPathGetTempPath : string
13732: Function jxPathIsAbsolute( const Path : string ) : Boolean
13733: Function jxPathIsChild( const Path, Base : string ) : Boolean
13734: Function jxPathIsDiskDevice( const Path : string ) : Boolean
13735: Function jxPathIsUNC( const Path : string ) : Boolean
13736: Function jxPathRemoveSeparator( const Path : string ) : string
13737: Function jxPathRemoveExtension( const Path : string ) : string
13738: Function jxPathGetPhysicalPath( const LocalizedPath : string ) : string
13739: Function jxPathGetLocalizedPath( const PhysicalPath : string ) : string
13740: JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders )
13741: JxTFileListOptions', 'set of TFileListOption
13742: JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13743: TFileHandler', 'Procedure ( const FileName : string )
13744: TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec )
13745: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings ) : Boolean
13746: //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
13747: AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
13748: FileMatchFunc:TFileMatchFunc):Bool;
13749: Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int ) : Boolean
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 jxFolderGetSize( const FileName : string ) : Int64
13775: Function jxFolderGetTempName( const Prefix : string ) : string
13776: Function jxFolderGetType( const FileName : string ) : string
13777: Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string ) : string
13778: Function jxFolderDirectories( Name : string ) : Boolean
13779: Function jxFolderDirectorySize( const Path : string ) : Int64
13780: Function jxFolderDriveTypeStr( const Drive : Char ) : string
13781: Function jxFolderGetAgeCoherence( const FileName : string ) : Boolean
13782: Procedure jxFolderAttributeList( const Items : TStrings; const Attr : Integer )
13783: Procedure jxFolderAttributeListEx( const Items : TStrings; const Attr : Integer )
13784: Function jxFolderInformation( const FileName : string; out FileInfo : TSearchRec ) : Boolean;
13785: Function jxFolderInformation1( const FileName : string ) : TSearchRec;
13786: //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
13787: ResolveSymLinks:Boolean):Integer

```

```

13787: Function jxGetFileLastWrite( const FName : string ) : TFileTime;
13788: Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13789: Function jxGetFileLastAccess( const FName : string ) : TFileTime;
13790: Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13791: Function jxGetFileCreation( const FName : string ) : TFileTime;
13792: Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13793: Function jxGetFileLastWrite( const FName : string; out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13794: Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13795: Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean ) : Integer;
13796: Function jxGetFileLastAccess( const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13797: Function jxGetFileLastAccess1( const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13798: Function jxGetFileLastAccess2( const FName:string; ResolveSymLinks:Boolean): Integer;
13799: Function jxGetFileLastAttrChange( const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13800: Function jxGetFileLastAttrChange1( const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Integer;
13801: Function jxGetModulePath( const Module : HMODULE ) : string;
13802: Function jxGetSizeOfFile( const FileName : string ) : Int64;
13803: Function jxGetSizeOfFile1( const FileInfo : TSearchRec ) : Int64;
13804: Function jxGetSizeOfFile2( Handle : THandle ) : Int64;
13805: Function jxGetStandardFileInfo( const FileName : string ) : TWin32FileAttributeData;
13807: Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean ) : Boolean;
13808: Function jxIsRootDirectory( const CanonicFileName : string ) : Boolean;
13809: Function jxLockVolume( const Volume : string; var Handle : THandle ) : Boolean;
13810: Function jxOpenVolume( const Drive : Char ) : THandle;
13811: Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime ) : Boolean;
13812: Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime ) : Boolean;
13813: Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime ) : Boolean;
13814: Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime ) : Boolean;
13815: Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime ) : Boolean;
13816: Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime ) : Boolean;
13817: Procedure jxShredfile( const FileName : string; Times : Integer );
13818: Function jxUnlockVolume( var Handle : THandle ) : Boolean;
13819: Function jxCreateSymbolicLink( const Name, Target : string ) : Boolean;
13820: Function jxSymbolicLinkTarget( const Name : string ) : string;
13821: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )';
13822: SIRegister_TJclCustomFileAttrMask(CL);
13823: SIRegister_TJclFileAttributeMask(CL);
13824: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS' +
13825: +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)';
13826: TFileSearchOptions', 'set of TFileSearchOption';
13827: TFileSearchTaskID', 'Integer;
13828: TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc' +
13829: +'hTaskID; const Aborted : Boolean)';
13830: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )';
13831: SIRegister_IJclFileEnumerator(CL);
13832: SIRegister_TJclFileVersionInfo(CL);
13833: Function JxFileVersion : IJclFileEnumerator;
13834: JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched, ffPreRelease, ffPrivateBuild, ffSpecialBuild )';
13835: JxTFileFlags', 'set of TFileFlag;
13836: FindClass('TOBJECT'), 'EJclFileVersionInfoError';
13837: SIRegister_TJclFileVersionInfo(CL);
13838: Function jxOSIdentToString( const OSIdent : DWORD ) : string;
13839: Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD ) : string;
13840: Function jxVersionResourceAvailable( const FileName : string ) : Boolean;
13841: TFileVersionFormat', '( vfMajorMinor, vfFull )';
13842: Function jxFormatVersionString( const HiV, LoV : Word ) : string;
13843: Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word ) : string;
13844: //Function FormatVersionString2( const FileInfo : TVSFixedFileInfo; VersionFormat:TFFileVersionFormat):str;
13845: //Procedure VersionExtractFileInfo(const FileInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13846: //Procedure VersionExtractProductInfo(const FileInfo:TVSFixedFileInfo;var Major,Minor,Build,
13847: Revision:Word);
13847: //Function VersionFixedFileInfo( const FileName : string; var FileInfo : TVSFixedFileInfo ) : Boolean;
13848: Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFFileVersionFormat; const
13849: NotAvailableText : string ) : string;
13849: SIRegister_TJclTempFileStream(CL);
13850: FindClass('TOBJECT'), 'TJclCustomFileMapping';
13851: SIRegister_TJclFileMapView(CL);
13852: TJclFileMappingRoundOffset', '( rvDown, rvUp )';
13853: SIRegister_TJclCustomFileMapping(CL);
13854: SIRegister_TJclFileMapping(CL);
13855: SIRegister_TJclSwapFileMapping(CL);
13856: SIRegister_TJclFileMappingStream(CL);
13857: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )';
13858: //PPCharArray', '^TPCharArray // will not work
13859: SIRegister_TJclMappedTextReader(CL);
13860: SIRegister_TJclFileMaskComparator(CL);
13861: FindClass('TOBJECT'), 'EJclPathError';
13862: FindClass('TOBJECT'), 'EJclFileUtilsError';
13863: FindClass('TOBJECT'), 'EJclTempFileStreamError';
13864: FindClass('TOBJECT'), 'EJclTempFileStreamError';
13865: FindClass('TOBJECT'), 'EJclFileMappingError';
13866: FindClass('TOBJECT'), 'EJclFileMappingViewError';
13867: Function jxPathGetLongName2( const Path : string ) : string;
13868: Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean;
13869: Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstfileName : string ) : Boolean;
13870: Function jxWin32BackupFile( const FileName : string; Move : Boolean ) : Boolean;
13871: Function jxWin32RestoreFile( const FileName : string ) : Boolean;
13872: Function jxSamePath( const Path1, Path2 : string ) : Boolean;
13873: Procedure jxPathListAddItems( var List : string; const Items : string );

```

```

13874: Procedure jxPathListIncludeItems( var List : string; const Items : string)
13875: Procedure jxPathListDeleteItems( var List : string; const Items : string)
13876: Procedure jxPathListDeleteItem( var List : string; const Index : Integer)
13877: Function jxPathListItemCount( const List : string) : Integer
13878: Function jxPathListGetItem( const List : string; const Index : Integer) : string
13879: Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13880: Function jxPathListItemIndex( const List, Item : string) : Integer
13881: Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13882: Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13883: Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
    AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13884: Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
    AllowedPrefixCharacters : string) : Integer
13885: end;
13886:
13887: procedure SIRRegister_FileUtil(CL: TPSCompiler);
13888: begin
13889:   'UTF8FileHeader','String #$ef#$bb#$bf';
13890:   Function lCompareFilenames( const Filenam1, Filenam2 : string) : integer
13891:   Function lCompareFilenamesIgnoreCase( const Filenam1, Filenam2 : string) : integer
13892:   Function lCompareFilenames( const Filenam1, Filenam2 : string; ResolveLinks : boolean) : integer
13893:   Function lCompareFilenames(Filenam1:PChar;Len1:int;Filenam2:PChar;Len2:int;ResolveLinks:boolean):int;
13894:   Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13895:   Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13896:   Function lFilenameIsUnixAbsolute( const Thefilename : string) : boolean
13897:   Procedure lCheckIfFileIsExecutable( const Afilename : string)
13898:   Procedure lCheckIfFileIsSymlink( const Afilename : string)
13899:   Function lFileIsReadable( const Afilename : string) : boolean
13900:   Function lFileIsWritable( const Afilename : string) : boolean
13901:   Function lFileIsText( const Afilename : string) : boolean
13902:   Function lFileIsText( const Afilename : string; out FileReadable : boolean) : boolean
13903:   Function lFileIsExecutable( const Afilename : string) : boolean
13904:   Function lFileIsSymlink( const Afilename : string) : boolean
13905:   Function lFileIsHardLink( const Afilename : string) : boolean
13906:   Function lFileSize( const Filenam : string) : int64;
13907:   Function lGetFileDescription( const Afilename : string) : string
13908:   Function lReadAllLinks( const Filenam : string; ExceptionOnError : boolean) : string
13909:   Function lTryReadAllLinks( const Filenam : string) : string
13910:   Function lDirPathExists( const FileName : String) : Boolean
13911:   Function lForceDirectory( DirectoryName : string) : boolean
13912:   Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13913:   Function lProgramDirectory : string
13914:   Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13915:   Function lExtractFileNameOnly( const Afilename : string) : string
13916:   Function lExtractFileNameWithoutExt( const Afilename : string) : string
13917:   Function lCompareFileExt( const Filenam, Ext : string; CaseSensitive : boolean) : integer;
13918:   Function lCompareFileExt( const Filenam, Ext : string) : integer;
13919:   Function lFilenameIsPascalUnit( const Filenam : string) : boolean
13920:   Function lAppendPathDelim( const Path : string) : string
13921:   Function lChompPathDelim( const Path : string) : string
13922:   Function lTrimFilename( const Afilename : string) : string
13923:   Function lCleanAndExpandFilename( const Filenam : string) : string
13924:   Function lCleanAndExpandDirectory( const Filenam : string) : string
13925:   Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13926:   Function lCreateRelativePath( const Filenam, BaseDirectory : string; UsePointDirectory : boolean;
    AlwaysRequireSharedBaseFolder : Boolean) : string
13927:   Function lCreateAbsolutePath( const Filenam, BaseDirectory : string) : string
13928:   Function lFileIsInPath( const Filenam, Path : string) : boolean
13929:   Function lFileIsInDirectory( const Filenam, Directory : string) : boolean
13930:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13931:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13932:   'AllDirectoryEntriesMask','String '*
13933:   Function l GetAllFilesMask : string
13934:   Function lGetExeExt : string
13935:   Function lSearchFileInPath( const Filenam, basePath, SearchPath, Delimiter : string; Flags :
    TSearchFileInPathFlags) : string
13936:   Function lSearchAllFilesInPath( const Filenam, basePath, SearchPath, Delimiter:string;Flags :
    TSearchFileInPathFlags) : TStrings
13937:   Function lFindDiskFilename( const Filenam : string) : string
13938:   Function lFindDiskFileCaseInsensitive( const Filenam : string) : string
13939:   Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13940:   Function lGetDarwinSystemFilename( Filenam : string) : string
13941:   SIRRegister_TFileIterator(CL);
13942:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13943:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13944:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13945:   SIRRegister_TFileSearcher(CL);
13946:   Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13947:   Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13948: // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13949: // TCopyFileFlags', 'set of TCopyFileFlag
13950:   Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13951:   Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13952:   Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13953:   Function lReadFileToString( const Filenam : string) : string
13954:   Function lGetTempFilename( const Directory, Prefix : string) : string
13955:   {Function NeedRTLAnsi : boolean
13956:   Procedure SetNeedRTLAnsi( NewValue : boolean)
13957:   Function UTF8ToSys( const s : string) : string

```

```

13958: Function SysToUTF8( const s : string ) : string
13959: Function ConsoleToUTF8( const s : string ) : string
13960: Function UTF8ToConsole( const s : string ) : string
13961: Function FileExistsUTF8( const FileName : string ) : boolean
13962: Function FileAgeUTF8( const FileName : string ) : Longint
13963: Function DirectoryExistsUTF8( const Directory : string ) : Boolean
13964: Function ExpandFileNameUTF8( const FileName : string ) : string
13965: Function ExpandUNCfileNameUTF8( const FileName : string ) : string
13966: Function ExtractShortPathNameUTF8( const FileName : String ) : String
13967: Function FindFirstUTF8( const Path: string; Attr : Longint; out Rslt : TSearchRec ) : Longint
13968: Function FindNextUTF8( var Rslt : TSearchRec ) : Longint
13969: Procedure FindCloseUTF8( var F : TSearchrec )
13970: Function FileSetDateUTF8( const FileName : String; Age : Longint ) : Longint
13971: Function FileGetAttrUTF8( const FileName : String ) : Longint
13972: Function FileSetAttrUTF8( const FileName : String; Attr : longint ) : Longint
13973: Function DeleteFileUTF8( const FileName : String ) : Boolean
13974: Function RenameFileUTF8( const OldName, NewName : String ) : Boolean
13975: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean ) : String
13976: Function FileIsReadOnlyUTF8( const FileName : String ) : Boolean
13977: Function GetCurrentDirUTF8 : String
13978: Function SetCurrentDirUTF8( const NewDir : String ) : Boolean
13979: Function CreateDirUTF8( const NewDir : String ) : Boolean
13980: Function RemoveDirUTF8( const Dir : String ) : Boolean
13981: Function ForceDirectoriesUTF8( const Dir : string ) : Boolean
13982: Function FileOpenUTF8( const FileName : string; Mode : Integer ) : THandle
13983: Function FileCreateUTF8( const FileName : string ) : THandle;
13984: Function FileCreateUTF8l( const FileName : string; Rights : Cardinal ) : THandle;
13985: Function ParamStrUTF8( Param : Integer ) : string
13986: Function GetEnvironmentStringUTF8( Index : Integer ) : string
13987: Function GetEnvironmentVariableUTF8( const EnvVar : string ) : String
13988: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean ) : string
13989: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean ) : string
13990: Function SysErrorMessageUTF8( ErrorCode : Integer ) : String
13991: end;
13992:
13993: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13994: begin
13995:   //VK_F23 = 134;
13996:   //{$EXTERNALSYM VK_F24}
13997:   //VK_F24 = 135;
13998:   TVirtualKeyCode', 'Integer
13999:   'VK_MOUSEWHEELUP', 'integer'(134);
14000:   'VK_MOUSEWHEELDOWN', 'integer'(135);
14001:   Function glIsKeyDown( c : Char ) : Boolean;
14002:   Function glIsKeyDown1( vk : TVirtualKeyCode ) : Boolean;
14003:   Function glKeyPressed( minVkeyCode : TVirtualKeyCode ) : TVirtualKeyCode
14004:   Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode ) : String
14005:   Function glKeyNameToVirtualKeyCode( const keyName : String ) : TVirtualKeyCode
14006:   Function glCharToVirtualKeyCode( c : Char ) : TVirtualKeyCode
14007:   Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer )
14008: end;
14009:
14010: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14011: begin
14012:   TGLPoint', 'TPoint
14013:   //PGLPoint', '^TGLPoint // will not work
14014:   TGLRect', 'TRect
14015:   //PGLRect', '^TGLRect // will not work
14016:   TDelphiColor', 'TColor
14017:   TGLPicture', 'TPicture
14018:   TGLGraphic', 'TGraphic
14019:   TGLBitmap', 'TBitmap
14020:   //TGraphicClass', 'class of TGraphic
14021:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14022:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
14023:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
14024:     + 'Button; Shift : TShiftState; X, Y : Integer )
14025:   TGLMouseMoveEvent', 'TMouseEvent
14026:   TGLKeyEvent', 'TKeyEvent
14027:   TGLKeyPressEvent', 'TKeyPressEvent
14028:   EGLOSError', 'EWin32Error
14029:   EGLOSError', 'EWin32Error
14030:   EGLOSError', 'EOSError
14031:   'gl$AllFilter', 'string'All // $AllFilter
14032:   Function GLPoint( const x, y : Integer ) : TGLPoint
14033:   Function GLRGB( const r, g, b : Byte ) : TColor
14034:   Function GLColorToRGB( color : TColor ) : TColor
14035:   Function GLGetRValue( rgb : DWORD ) : Byte
14036:   Function GLGetGValue( rgb : DWORD ) : Byte
14037:   Function GLGetBValue( rgb : DWORD ) : Byte
14038:   Procedure GLInitWinColors
14039:   Function GLRect( const aLeft, aTop, aRight, aBottom : Integer ) : TGLRect
14040:   Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer )
14041:   Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect )
14042:   Procedure GLInformationDlg( const msg : String )
14043:   Function GLQuestionDlg( const msg : String ) : Boolean
14044:   Function GLInputDlg( const msg : String ) : String
14045:   Function GLSavePictureDialog( var aFileName : String; const aTitle : String ) : Boolean
14046:   Function GLOpenPictureDialog( var aFileName : String; const aTitle : String ) : Boolean

```

```

14047: Function GLApplicationTerminated : Boolean
14048: Procedure GLRaiseLastOSError
14049: Procedure GLFreeAndNil( var anObject: TObject)
14050: Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
14051: Function GLGetCurrentColorDepth : Integer
14052: Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14053: Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14054: Procedure GLSleep( length : Cardinal)
14055: Procedure GLQueryPerformanceCounter( var val : Int64)
14056: Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14057: Function GLStartPrecisionTimer : Int64
14058: Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14059: Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14060: Function GLRTSC : Int64
14061: Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14062: Function GLOKMessageBox( const Text, Caption : string) : Integer
14063: Procedure GLShowHTMLUrl( Url : String)
14064: Procedure GLShowCursor( AShow : boolean)
14065: Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14066: Procedure GLGetCursorPos( var point : TGLPoint)
14067: Function GLGetScreenWidth : integer
14068: Function GLGetScreenHeight : integer
14069: Function GLGetTickCount : int64
14070: function RemoveSpaces(const str : String) : String;
14071: TNoramlMapSpace', ' nmsObject, nmsTangent )
14072: Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList:Colors:TVectorList)
14073: Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14074: Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals : TAffineVectorList; Colors : TVectorList)
14075: Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
HiTexCoords:TAffineVectorList):TGLBitmap
14076: Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14077: end;
14078:
14079: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14080: begin
14081:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14082: // PGLStarRecord', '^TGLStarRecord // will not work
14083: Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14084: Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14085: Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14086: end;
14087:
14088:
14089: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14090: begin
14091:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14092: //PAABB', '^TAABB // will not work
14093: TBSphere', 'record Center : TAffineVector; Radius : single; end
14094: TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14095: TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14096: Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox ) : THmgBoundingBox
14097: Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14098: Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14099: Procedure SetAABB( var bb : TAABB; const v : TVector)
14100: Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14101: Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14102: Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14103: Function BBMinX( const c : THmgBoundingBox ) : Single
14104: Function BBMaxX( const c : THmgBoundingBox ) : Single
14105: Function BBMinY( const c : THmgBoundingBox ) : Single
14106: Function BBMaxY( const c : THmgBoundingBox ) : Single
14107: Function BBMinZ( const c : THmgBoundingBox ) : Single
14108: Function BBMaxZ( const c : THmgBoundingBox ) : Single
14109: Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14110: Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14111: Function AABBIntersection( const aabb1, aabb2 : TAABB ) : TAABB
14112: Function BBTAAABB( const aBB : THmgBoundingBox ) : TAABB
14113: Function AABBTaaBB( const anAABB : TAABB ) : THmgBoundingBox;
14114: Function AABBTaaB1( const anAABB : TAABB; const m : TMatrix ) : THmgBoundingBox;
14115: Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14116: Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14117: Function IntersectAABBS( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix ) : Boolean;
14118: Function IntersectAABBSAbsoluteXY( const aabb1, aabb2 : TAABB ) : Boolean
14119: Function IntersectAABBSAbsoluteXZ( const aabb1, aabb2 : TAABB ) : Boolean
14120: Function IntersectAABBSAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14121: Function AABBFitInAABBAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14122: Function PointInAABB( const p : TAffineVector; const aabb : TAABB ) : Boolean;
14123: Function PointInAABB1( const p : TVector; const aabb : TAABB ) : Boolean;
14124: Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB ) : boolean
14125: Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector ) : boolean
14126: Procedure ExtractAABCorners( const AABB : TAABB; var AABCorners : TAABCorners)
14127: Procedure AABBTaaBSphere( const AABB : TAABB; var BSphere : TBSphere)
14128: Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14129: Function BSphereToAABB1( const center : TAffineVector; radius : Single ) : TAABB;
14130: Function BSphereToAABB2( const center : TVector; radius : Single ) : TAABB;
14131: Function AABBContainsAABB( const mainAABB, testAABB : TAABB ) : TSpaceContains
14132: Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB ) : TSpaceContains

```

```

14133: Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere ) : TSpaceContains
14134: Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere ) : TSpaceContains
14135: Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
testBSphere:TBSphere):TSpaceContains
14136: Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere ) : TSpaceContains
14137: Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB ) : TSpaceContains
14138: Function ClipToAABB( const v : TAffineVector; const AABB : TAABB ) : TAffineVector
14139: Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere ) : boolean
14140: Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single )
14141: Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
viewportSizeY:Int):TClipRect
14142: end;
14143:
14144: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14145: begin
14146: Procedure Cylindrical_Cartesian( const r, theta, zl : single; var x, y, z : single );
14147: Procedure Cylindrical_Cartesian1( const r, theta, zl : double; var x, y, z : double );
14148: Procedure Cylindrical_Cartesian2( const r,theta,zl : single; var x, y, z : single; var ierr : integer );
14149: Procedure Cylindrical_Cartesian3( const r,theta,zl : double; var x, y, z : double; var ierr : integer );
14150: Procedure Cartesian_Cylindrical( const x, y, zl : single; var r, theta, z : single );
14151: Procedure Cartesian_Cylindrical1( const x, y, zl : double; var r, theta, z : double );
14152: Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single );
14153: Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double );
14154: Procedure Spherical_Cartesian2( const r,theta,phi : single; var x, y, z : single; var ierr : integer );
14155: Procedure Spherical_Cartesian3( const r,theta,phi : double; var x, y, z : double; var ierr : integer );
14156: Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single );
14157: Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single );
14158: Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double );
14159: Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single );
14160: Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double );
14161: Procedure ProlateSpheroidal_Cartesian2( const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer );
14162: Procedure ProlateSpheroidal_Cartesian3( const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer );
14163: Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single );
14164: Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double );
14165: Procedure OblateSpheroidal_Cartesian2( const xi,eta,phi,a:single;var x,y,z:single;var ierr:integer );
14166: Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double;var x,y,z:double;var ierr:integer );
14167: Procedure BipolarCylindrical_Cartesian( const u, v, zl, a : single; var x, y, z : single );
14168: Procedure BipolarCylindrical_Cartesian1( const u, v, zl, a : double; var x, y, z : double );
14169: Procedure BipolarCylindrical_Cartesian2( const u,v,zl,a : single;var x,y,z:single; var ierr : integer );
14170: Procedure BipolarCylindrical_Cartesian3( const u,v,zl,a : double;var x,y,z:double; var ierr : integer );
14171: end;
14172:
14173: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14174: begin
14175: 'EPSILON','Single').setExtended( 1e-40 );
14176: 'EPSILON2','Single').setExtended( 1e-30 ); }
14177: TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14178: + 'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14179: THmgPlane', 'TVector
14180: TDoubleHmgPlane', 'THomogeneousDblVector
14181: {TTtransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14182: +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14183: +' , ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14184: TSingleArray', 'array of Single
14185: TTtransformations', 'array [0..15] of Single)
14186: TPackedRotationMatrix', 'array [0..2] of Smallint)
14187: TVertex', 'TAffineVector
14188: //TVectorGL', 'THomogeneousFltVector
14189: //TMatrixGL', 'THomogeneousFltMatrix
14190: // TPackedRotationMatrix = array [0..2] of SmallInt;
14191: Function glTexPointMake( const s, t : Single ) : TTexPoint
14192: Function glAffineVectorMake( const x, y, z : Single ) : TAffineVector;
14193: Function glAffineVectorMake1( const v : TVectorGL ) : TAffineVector;
14194: Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single );
14195: Procedure glSetVector( var v : TAffineVector; const x, y, z : Single );
14196: Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL );
14197: Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector );
14198: Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector );
14199: Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL );
14200: Function glVectorMake( const v : TAffineVector; w : Single ) : TVectorGL;
14201: Function glVectorMake1( const x, y, z : Single; w : Single ) : TVectorGL;
14202: Function glPointMake( const x, y, z : Single ) : TVectorGL;
14203: Function glPointMake1( const v : TAffineVector ) : TVectorGL;
14204: Function glPointMake2( const v : TVectorGL ) : TVectorGL;
14205: Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single );
14206: Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single );
14207: Procedure glg1SetVector7( var v : TVectorGL; const vSrc : TVectorGL );
14208: Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single );
14209: Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector );
14210: Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL );
14211: Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single );
14212: Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single );
14213: Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector );
14214: Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL );
14215: Procedure glRstVector( var v : TAffineVector );
14216: Procedure glRstVector1( var v : TVectorGL );
14217: Function glVectorAdd( const v1, v2 : TAffineVector ) : TAffineVector;
14218: Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector );
14219: //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector );

```

```

14220: Function glVectorAdd3( const v1, v2 : TVectorGL ) : TVectorGL;
14221: Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL );
14222: Function glVectorAdd5( const v : TAffineVector; const f : Single ) : TAffineVector;
14223: Function glVectorAdd6( const v : TVectorGL; const f : Single ) : TVectorGL;
14224: Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector );
14225: Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL );
14226: Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL );
14227: Procedure glAddVector10( var v : TAffineVector; const f : Single );
14228: Procedure glAddVector11( var v : TVectorGL; const f : Single );
14229: //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Int;dest:PTexPointArray);
14230: //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Integer;const scale:TTexPoint; dest : PTExPointArray);
14231: //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
PAffineVectorArray);
14232: Function glVectorSubtract( const V1, V2 : TAffineVector ) : TAffineVector;
14233: Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector );
14234: Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL );
14235: Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL );
14236: Function glVectorSubtract4( const V1, V2 : TVectorGL ) : TVectorGL;
14237: Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL );
14238: Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector );
14239: Function glVectorSubtract7( const v1 : TAffineVector; delta : Single ) : TAffineVector;
14240: Function glVectorSubtract8( const v1 : TVectorGL; delta : Single ) : TVectorGL;
14241: Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector );
14242: Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL );
14243: Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single );
14244: //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat );
14245: Function glTexPointCombine( const t1, t2 : TTExPoint; f1, f2 : Single ) : TTExPoint;
14246: Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single ) : TAffineVector;
14247: Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single ) : TAffineVector;
14248: Procedure glVectorCombine34( const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector );
14249: Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single );
14250: Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single );
14251: Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single ) : TVectorGL;
14252: Function glVectorCombine8( const V1 : TVectorGL; const V2 : TAffineVector; const F1,F2:Single ) : TVectorGL;
14253: Procedure glVectorCombine9( const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL );
14254: Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL );
14255: Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL );
14256: Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single ) : TVectorGL;
14257: Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL );
14258: Function glVectorDotProduct( const V1, V2 : TAffineVector ) : Single;
14259: Function glVectorDotProduct1( const V1, V2 : TVectorGL ) : Single;
14260: Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector ) : Single;
14261: Function glPointProject( const p, origin, direction : TAffineVector ) : Single;
14262: Function glPointProject1( const p, origin, direction : TVectorGL ) : Single;
14263: Function glVectorCrossProduct( const V1, V2 : TAffineVector ) : TAffineVector;
14264: Function glVectorCrossProduct1( const V1, V2 : TVectorGL ) : TVectorGL;
14265: Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL );
14266: Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL );
14267: Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector );
14268: Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector );
14269: Function glLerp( const start, stop, t : Single ) : Single;
14270: Function glAngleLerp( start, stop, t : Single ) : Single;
14271: Function glDistanceBetweenAngles( angle1, angle2 : Single ) : Single;
14272: Function glTexPointLerp( const t1, t2 : TTExPoint; t : Single ) : TTExPoint;
14273: Function glVectorLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14274: Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector );
14275: Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single ) : TVectorGL;
14276: Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL );
14277: Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14278: Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single ) : TAffineVector;
14279: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray );
14280: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
PAffineVectorArray );
14281: Function glVectorLength( const x, y : Single ) : Single;
14282: Function glVectorLength1( const x, y, z : Single ) : Single;
14283: Function glVectorLength2( const v : TAffineVector ) : Single;
14284: Function glVectorLength3( const v : TVectorGL ) : Single;
14285: Function glVectorLength4( const v : array of Single ) : Single;
14286: Function glVectorNorm( const x, y : Single ) : Single;
14287: Function glVectorNorm1( const v : TAffineVector ) : Single;
14288: Function glVectorNorm2( const v : TVectorGL ) : Single;
14289: Function glVectorNorm3( var V : array of Single ) : Single;
14290: Procedure glNormalizeVector( var v : TAffineVector );
14291: Procedure glNormalizeVector1( var v : TVectorGL );
14292: Function glVectorNormalize( const v : TAffineVector ) : TAffineVector;
14293: Function glVectorNormalize1( const v : TVectorGL ) : TVectorGL;
14294: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer );
14295: Function glVectorAngleCosine( const V1, V2 : TAffineVector ) : Single;
14296: Function glVectorNegate( const v : TAffineVector ) : TAffineVector;
14297: Function glVectorNegate1( const v : TVectorGL ) : TVectorGL;
14298: Procedure glNegateVector( var V : TAffineVector );
14299: Procedure glNegateVector2( var V : TVectorGL );
14300: Procedure glNegateVector3( var V : array of Single );
14301: Procedure glScaleVector( var v : TAffineVector; factor : Single );
14302: Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector );
14303: Procedure glScaleVector2( var v : TVectorGL; factor : Single );
14304: Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL );

```

```

14305: Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14306: Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14307: Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14308: Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14309: Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14310: Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14311: Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14312: Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14313: Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14314: Function glVectorIsNull( const v : TVectorGL) : Boolean;
14315: Function glVectorIsNotNull( const v : TAffineVector) : Boolean;
14316: Function glVectorSpacing( const v1, v2 : TTExPoint) : Single;
14317: Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14318: Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14319: Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14320: Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14321: Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14322: Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14323: Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector;
14324: Function glVectorReflect( const V, N : TAffineVector) : TAffineVector;
14325: Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14326: Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14327: Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single);
14328: Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14329: Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14330: Procedure glVectorRotateAroundY1( const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14331: Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14332: Procedure glAbsVector( var v : TVectorGL);
14333: Procedure glAbsVector1( var v : TAffineVector);
14334: Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14335: Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14336: Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14337: Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14338: Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14339: Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14340: Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14341: Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14342: Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14343: Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14344: Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14345: Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14346: Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14347: Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14348: Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14349: Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14350: Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14351: Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14352: Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14353: Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix;
14354: Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14355: Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14356: Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14357: Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14358: Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14359: Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14360: Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14361: Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14362: Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14363: Procedure glAdjointMatrix( var M : TMatrixGL);
14364: Procedure glAdjointMatrix1( var M : TAffineMatrix);
14365: Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14366: Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14367: Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14368: Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14369: Procedure glNormalizeMatrix( var M : TMatrixGL);
14370: Procedure glTransposeMatrix( var M : TAffineMatrix);
14371: Procedure glTransposeMatrix1( var M : TMatrixGL);
14372: Procedure glInvertMatrix( var M : TMatrixGL);
14373: Procedure glInvertMatrix1( var M : TAffineMatrix);
14374: Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL;
14375: Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean;
14376: Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14377: Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14378: Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14379: Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14380: Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane);
14381: Procedure glNormalizePlane( var plane : THmgPlane);
14382: Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14383: Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14384: Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14385: Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14386: Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14387: Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14388: Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14389: Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14390: Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14391: Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector;
14392: Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single;
14393: Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector

```

```

14394: Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector ) : Single
14395: Procedure SgsegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
Segment0Closest, Segment1Closest : TAffineVector)
14396: Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14397: TEulerOrder', (' eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14398: Function glQuaternionMake( const Imag : array of Single; Real : Single ) : TQuaternion
14399: Function glQuaternionConjugate( const Q : TQuaternion ) : TQuaternion
14400: Function glQuaternionMagnitude( const Q : TQuaternion ) : Single
14401: Procedure glNormalizeQuaternion( var Q : TQuaternion )
14402: Function glQuaternionFromPoints( const V1, V2 : TAffineVector ) : TQuaternion
14403: Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector )
14404: Function glQuaternionFromMatrix( const mat : TMatrixGL ) : TQuaternion
14405: Function glQuaternionToMatrix( quat : TQuaternion ) : TMatrixGL
14406: Function glQuaternionToAffineMatrix( quat : TQuaternion ) : TAffineMatrix
14407: Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector ) : TQuaternion
14408: Function glQuaternionFromRollPitchYaw( const r, p, y : Single ) : TQuaternion
14409: Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder ) : TQuaternion
14410: Function glQuaternionMultiply( const qL, qR : TQuaternion ) : TQuaternion
14411: Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single ) : TQuaternion;
14412: Function glQuaternionSlerpl( const source, dest : TQuaternion; const t : Single ) : TQuaternion;
14413: Function glLnXp1( X : Extended ) : Extended
14414: Function glLog10( X : Extended ) : Extended
14415: Function glLog2( X : Extended ) : Extended;
14416: Function glLog2l( X : Single ) : Single;
14417: Function glLogN( Base, X : Extended ) : Extended
14418: Function glIntPower( Base : Extended; Exponent : Integer ) : Extended
14419: Function glPower( const Base, Exponent : Single ) : Single;
14420: Function glPowerl( Base : Single; Exponent : Integer ) : Single;
14421: Function glDegToRad( const Degrees : Extended ) : Extended;
14422: Function glDegToRadl( const Degrees : Single ) : Single;
14423: Function glRadToDeg( const Radians : Extended ) : Extended;
14424: Function glRadToDegl( const Radians : Single ) : Single;
14425: Function glNormalizeAngle( angle : Single ) : Single
14426: Function glNormalizeDegAngle( angle : Single ) : Single
14427: Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended );
14428: Procedure glSinCosl( const Theta : Double; var Sin, Cos : Double );
14429: Procedure glSinCos( const Theta : Single; var Sin, Cos : Single );
14430: Procedure glSinCosl( const theta, radius : Double; var Sin, Cos : Extended );
14431: Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double );
14432: Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single );
14433: Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single )
14434: Function glArcCos( const X : Extended ) : Extended;
14435: Function glArcCosl( const x : Single ) : Single;
14436: Function glArcSin( const X : Extended ) : Extended;
14437: Function glArcSinl( const X : Single ) : Single;
14438: Function glArcTan2l( const Y, X : Extended ) : Extended;
14439: Function glArcTan2l( const Y, X : Single ) : Single;
14440: Function glFastArcTan2( y, x : Single ) : Single
14441: Function glTan( const X : Extended ) : Extended;
14442: Function glTanl( const X : Single ) : Single;
14443: Function glCoTan( const X : Extended ) : Extended;
14444: Function glCoTanol( const X : Single ) : Single;
14445: Function glSinh( const x : Single ) : Single;
14446: Function glSinhl( const x : Double ) : Double;
14447: Function glCosh( const x : Single ) : Single;
14448: Function glCoshl( const x : Double ) : Double;
14449: Function glRSqrt( v : Single ) : Single
14450: Function glRLength( x, y : Single ) : Single
14451: Function glISqrt( i : Integer ) : Integer
14452: Function glILength( x, y : Integer ) : Integer;
14453: Function glILengthl( x, y, z : Integer ) : Integer;
14454: Procedure glRegisterBasedExp
14455: Procedure glRandomPointOnSphere( var p : TAffineVector )
14456: Function glRoundInt( v : Single ) : Single;
14457: Function glRoundIntl( v : Extended ) : Extended;
14458: Function glTrunc( v : Single ) : Integer;
14459: Function glTrunc64( v : Extended ) : Int64;
14460: Function glInt( v : Single ) : Single;
14461: Function glIntl( v : Extended ) : Extended;
14462: Function glFrac( v : Single ) : Single;
14463: Function glFracl( v : Extended ) : Extended;
14464: Function glRound( v : Single ) : Integer;
14465: Function glRound64( v : Single ) : Int64;
14466: Function glRound64l( v : Extended ) : Int64;
14467: Function glTrunc( X : Extended ) : Int64
14468: Function glRound( X : Extended ) : Int64
14469: Function glFrac( X : Extended ) : Extended
14470: Function glCeil( v : Single ) : Integer;
14471: Function glCeil64( v : Extended ) : Int64;
14472: Function glFloor( v : Single ) : Integer;
14473: Function glFloor64( v : Extended ) : Int64;
14474: Function glScaleAndRound( i : Integer; var s : Single ) : Integer
14475: Function glSign( x : Single ) : Integer
14476: Function glIsInRange( const x, a, b : Single ) : Boolean;
14477: Function glIsInRangel( const x, a, b : Double ) : Boolean;
14478: Function glIsInCube( const p, d : TAffineVector ) : Boolean;
14479: Function glIsInCubel( const p, d : TVectorGL ) : Boolean;
14480: //Function MinFloat( values : PSingleArray; nbItems : Integer ) : Single;
14481: //Function MinFloat1( values : PDoubleArray; nbItems : Integer ) : Double;

```

```

14482: //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14483: Function glMinFloat3( const v1, v2 : Single) : Single;
14484: Function glMinFloat4( const v : array of Single) : Single;
14485: Function glMinFloat5( const v1, v2 : Double) : Double;
14486: Function glMinFloat6( const v1, v2 : Extended) : Extended;
14487: Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14488: Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14489: Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14490: //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14491: //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14492: //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14493: Function glMaxFloat2( const v : array of Single) : Single;
14494: Function glMaxFloat3( const v1, v2 : Single) : Single;
14495: Function glMaxFloat4( const v1, v2 : Double) : Double;
14496: Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14497: Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14498: Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14499: Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14500: Function glMinInteger9( const v1, v2 : Integer) : Integer;
14501: Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14502: Function glMaxInteger( const v1, v2 : Integer) : Integer;
14503: Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14504: Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14505: //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14506: Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14507: //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14508: //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14509: Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14510: //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14511: Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14512: //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14513: Function glMaxXYZComponent( const v : TVectorGL) : Single;
14514: Function glMaxXYZComponent1( const v : TAffineVector) : single;
14515: Function glMinXYZComponent( const v : TVectorGL) : Single;
14516: Function glMinXYZComponent1( const v : TAffineVector) : single;
14517: Function glMaxAbsXYZComponent( v : TVectorGL) : Single;
14518: Function glMinAbsXYZComponent( v : TVectorGL) : Single;
14519: Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14520: Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14521: Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14522: Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14523: Procedure glSortArrayAscending( var a : array of Extended);
14524: Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14525: Function glClampValue1( const aValue, aMin : Single) : Single;
14526: Function glGeometryOptimizationMode : String;
14527: Procedure glBeginFPUOnlySection;
14528: Procedure glEndFPUOnlySection;
14529: Function glConvertRotation( const Angles : TAffineVector) : TVectorGL;
14530: Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector;
14531: Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector;
14532: Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector;
14533: Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector;
14534: Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector;
14535: Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector;
14536: Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean;
14537: Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word);
14538: Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14539: Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14540: Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14541: Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14542: Function glRoll( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14543: Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14544: Function glRayCastMinDistToPoint( const rayStart, rayVector : TVectorGL;const point:TVectorGL):Single;
14545: Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14546: Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14547: Function glSphereVisibleRadius( distance, radius : Single) : Single;
14548: Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum;
14549: Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci : TRenderContextClippingInfo) : Boolean;
14550: Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci : TRenderContextClippingInfo) : Boolean;
14551: Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14552: Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
Frustum:TFrustum):Bool;
14553: Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL;
14554: Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL;
14555: Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL;
14556: Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix;
14557: Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL;
14558: 'cPI','Single').setExtended( 3.141592654);
14559: 'cPIdiv180','Single').setExtended( 0.017453292);
14560: 'c180divPI','Single').setExtended( 57.29577951);
14561: 'c2PI','Single').setExtended( 6.283185307);
14562: 'cPIdiv2','Single').setExtended( 1.570796326);
14563: 'cPIdiv4','Single').setExtended( 0.785398163);
14564: 'c3PIdiv4','Single').setExtended( 2.35619449);
14565: 'cInv2PI','Single').setExtended( 1 / 6.283185307);

```

```

14566: 'cInv360','Single').setExtended( 1 / 360);
14567: 'c180','Single').setExtended( 180);
14568: 'c360','Single').setExtended( 360);
14569: 'cOneHalf','Single').setExtended( 0.5);
14570: 'cLn10','Single').setExtended( 2.302585093);
14571: {'MinSingle','Extended').setExtended( 1.5e-45);
14572: 'MaxSingle','Extended').setExtended( 3.4e+38);
14573: 'MinDouble','Extended').setExtended( 5.0e-324);
14574: 'MaxDouble','Extended').setExtended( 1.7e+308);
14575: 'MinExtended','Extended').setExtended( 3.4e-4932);
14576: 'MaxExtended','Extended').setExtended( 1.1e+4932);
14577: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14578: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14579: end;
14580:
14581: procedure SIRегистер_GLVectorFileObjects(CL: TPSpascalCompiler);
14582: begin
14583:   AddClassN(FindClass('TOBJECT'), 'TMeshObjectList'
14584:     (FindClass('TOBJECT'), 'TFaceGroups
14585:      TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14586:      TMeshAutoCenterings', 'set of TMeshAutoCentering
14587:      TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14588:      SIRегистер_TBaseMeshObject(CL);
14589:      (FindClass('TOBJECT'), 'TSkeletonFrameList
14590:        TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14591:        SIRегистер_TSkeletonFrame(CL);
14592:        SIRегистер_TSkeletonFrameList(CL);
14593:        (FindClass('TOBJECT'), 'TSkeleton
14594:          (FindClass('TOBJECT'), 'TSkeletonBone
14595:          SIRегистер_TSkeletonBoneList(CL);
14596:          SIRегистер_TSkeletonRootBoneList(CL);
14597:          SIRегистер_TSkeletonBone(CL);
14598:          (FindClass('TOBJECT'), 'TSkeletonColliderList
14599:          SIRегистер_TSkeletonCollider(CL);
14600:          SIRегистер_TSkeletonColliderList(CL);
14601:          (FindClass('TOBJECT'), 'TGLBaseMesh
14602:            TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14603:              +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14604:              +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14605:              +'QuaternionList; end
14606:            SIRегистер_TSkeleton(CL);
14607:            TMeshObjectRenderingOption', '( moroGroupByMaterial )
14608:            TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14609:            SIRегистер_TMoshObject(CL);
14610:            SIRегистер_TMoshObjectList(CL);
14611: //TMeshObjectListClass', 'class of TMeshObjectList
14612: (FindClass('TOBJECT'), 'TMeshMorphTargetList
14613: SIRегистер_TMoshMorphTarget(CL);
14614: SIRегистер_TMoshMorphTargetList(CL);
14615: SIRегистер_TMorphableMeshObject(CL);
14616: TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14617: //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14618: //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14619: TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14620: SIRегистер_TSkeletonMeshObject(CL);
14621: SIRегистер_TFaceGroup(CL);
14622: TFaceGroupMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14623:   +'atTriangles, fgmmTriangleFan, fgmmQuads )
14624: SIRегистер_TFGVertexIndexList(CL);
14625: SIRегистер_TFGVertexNormalTexIndexList(CL);
14626: SIRегистер_TFGIndexTexCoordList(CL);
14627: SIRегистер_TFaceGroups(CL);
14628: TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14629: SIRегистер_TVectorFile(CL);
14630: //TVectorFileClass', 'class of TVectorFile
14631: SIRегистер_TGLGLSMVectorFile(CL);
14632: SIRегистер_TGLBaseMesh(CL);
14633: SIRегистер_TGLFreeForm(CL);
14634: TGLActorOption', '( aoSkeletonNormalizeNormals )
14635: TGLActorOptions', 'set of TGLActorOption
14636: 'cDefaultGLActorOptions', 'LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14637: (FindClass('TOBJECT'), 'TGLActor
14638: TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14639: SIRегистер_TActorAnimation(CL);
14640: TActorAnimationName', 'String
14641: SIRегистер_TActorAnimations(CL);
14642: SIRегистер_TGLBaseAnimationController(CL);
14643: SIRегистер_TGLAnimationController(CL);
14644: TActorFrameInterpolation', '( afpNone, afpLinear )
14645: TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc
14646:   +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14647: SIRегистер_TGLActor(CL);
14648: SIRегистер_TVectorFileFormat(CL);
14649: SIRегистер_TVectorFileFormatsList(CL);
14650: (FindClass('TOBJECT'), 'EInvalidVectorFile
14651: Function GetVectorFileFormats : TVectorFileFormatsList
14652: Function VectorFileFormatsFilter : String
14653: Function VectorFileFormatsSaveFilter : String
14654: Function VectorFileFormatExtensionByIndex( index : Integer ) : String

```

```

14655: Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14656: Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14657: end;
14658:
14659: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14660: begin
14661:   'Class_DColorPropPage', 'TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14662:   'Class_DFontPropPage', 'TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14663:   'Class_DPicturePropPage', 'TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14664:   'Class_DStringPropPage', 'TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14665:   SIRegister_TOLEStream(CL);
14666:   (FindClass('TOBJECT'), TConnectionPoints
14667:    ConnectionKind', '( ckSingle, ckMulti )
14668:   SIRegister_TConnectionPoint(CL);
14669:   SIRegister_TConnectionPoints(CL);
14670:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14671:   (FindClass('TOBJECT'), TActiveXControlFactory
14672:   SIRegister_TActiveXControl(CL);
14673:   //TActiveXControlClass', 'class of TActiveXControl
14674:   SIRegister_TActiveXControlFactory(CL);
14675:   SIRegister_TActiveFormControl(CL);
14676:   SIRegister_TActiveForm(CL);
14677:   //TActiveFormClass', 'class of TActiveForm
14678:   SIRegister_TActiveFormFactory(CL);
14679:   (FindClass('TOBJECT'), TPropertyPageImpl
14680:   SIRegister_TPropertyPage(CL);
14681:   //TPropertyPageClass', 'class of TPropertyPage
14682:   SIRegister_TPropertyPageImpl(CL);
14683:   SIRegister_TActiveXPropertyPage(CL);
14684:   SIRegister_TActiveXPropertyPageFactory(CL);
14685:   SIRegister_TCustomAdapter(CL);
14686:   SIRegister_TAdapterNotifier(CL);
14687:   SIRegister_TFontAccess(CL);
14688:   SIRegister_TFontAdapter(CL);
14689:   SIRegister_TPictureAccess(CL);
14690:   SIRegister_TPictureAdapter(CL);
14691:   SIRegister_TOLEGraphic(CL);
14692:   SIRegister_TStringsAdapter(CL);
14693:   SIRegister_TReflectorWindow(CL);
14694: Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:GUID;VTCode:Int;PropList:TStrings);
14695: Procedure GetOLEFont( Font : TFont; var OleFont : IFontDisp)
14696: Procedure SetOLEFont( Font : TFont; OleFont : IFontDisp)
14697: Procedure GetOLEPicture( Picture : TPicture; var OlePicture : IPictureDisp)
14698: Procedure SetOLEPicture( Picture : TPicture; OlePicture : IPictureDisp)
14699: Procedure GetOLEStrings( Strings : TStrings; var OleStrings : IStrings)
14700: Procedure SetOLEStrings( Strings : TStrings; OleStrings : IStrings)
14701: Function ParkingWindow : Hwnd
14702: end;
14703:
14704: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14705: begin
14706:   // TIP6Bytes = array [0..15] of Byte;
14707:   {binary form of IPv6 adress (for string conversion routines)}
14708:   // TIP6Words = array [0..7] of Word;
14709:   AddTypeS('TIP6Bytes', 'array [0..15] of Byte;');
14710:   AddTypeS('TIP6Words', 'array [0..7] of Word;');
14711:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4 : Byte; end');
14712:   Function synaIsIP( const Value : string) : Boolean';
14713:   Function synaIsIP6( const Value : string) : Boolean';
14714:   Function synaIPtoID( Host : string) : Ansistring';
14715:   Function synaStrToIp6( value : string) : TIP6Bytes';
14716:   Function synaIp6ToStr( value : TIP6Bytes) : string';
14717:   Function synaStrToIp( value : string) : integer';
14718:   Function synaIpToStr( value : integer) : string';
14719:   Function synaReverseIP( Value : AnsiString) : AnsiString';
14720:   Function synaReverseIP6( Value : AnsiString) : AnsiString';
14721:   Function synaExpandIP6( Value : AnsiString) : AnsiString';
14722:   Function xStrToIP( const Value : string) : TIPAdr';
14723:   Function xIPToStr( const Adresse : TIPAdr) : string';
14724:   Function IPToCardinal( const Adresse : TIPAdr) : Cardinal';
14725:   Function CardinalToIP( const Value : Cardinal) : TIPAdr';
14726: end;
14727:
14728: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14729: begin
14730:   AddTypeS('TSpecials', 'set of Char');
14731:   Const ('SpecialChar', 'TSpecials').SetSet( '='[]<>;@/?\'' );
14732:   Const ('URLFullSpecialChar', 'TSpecials').SetSet( '/?:@=&#+' );
14733:   Const ('TableBase64'ABCDEFIGHJKLMNOPQRSTUVWXYZZabcdefg hijklmnopqrstuvwxyz0123456789+/=+');
14734:   Const ('TableBase64mod'ABCDEFIGHJKLMNOPQRSTUVWXYZZabcdefg hijklmnopqrstuvwxyz0123456789+,=+');
14735:   Const ('TableUU'(`!#$%`)*+,-./0123456789:+<>?@ABCDEFIGHJKLMNOPQRSTUVWXYZ[\]^_');
14736:   Const ('TableXX'(+0123456789ABCDEFIGHJKLMNOPQRSTUVWXYZZabcdefg hijklmnopqrstuvwxyz');
14737:   Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString';
14738:   Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString';
14739:   Function DecodeURL( const Value : AnsiString) : AnsiString';
14740:   Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString';
14741:   Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString';
14742:   Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString';
14743:   Function EncodeURLElement( const Value : AnsiString) : AnsiString';

```

```

14744: Function EncodeURL( const Value : AnsiString ) : AnsiString');
14745: Function Decode4to3( const Value, Table : AnsiString ) : AnsiString');
14746: Function Decode4to3Ex( const Value, Table : AnsiString ) : AnsiString');
14747: Function Encode3to4( const Value, Table : AnsiString ) : AnsiString');
14748: Function synDecodeBase64( const Value : AnsiString ) : AnsiString');
14749: Function synEncodeBase64( const Value : AnsiString ) : AnsiString');
14750: Function DecodeBase64mod( const Value : AnsiString ) : AnsiString');
14751: Function EncodeBase64mod( const Value : AnsiString ) : AnsiString');
14752: Function DecodeUU( const Value : AnsiString ) : AnsiString');
14753: Function EncodeUU( const Value : AnsiString ) : AnsiString');
14754: Function DecodeXX( const Value : AnsiString ) : AnsiString');
14755: Function DecodeYEnc( const Value : AnsiString ) : AnsiString');
14756: Function UpdateCrc32( Value : Byte; Crc32 : Integer ) : Integer');
14757: Function synCrc32( const Value : AnsiString ) : Integer');
14758: Function UpdateCrc16( Value : Byte; Crc16 : Word ) : Word');
14759: Function Crc16( const Value : AnsiString ) : Word');
14760: Function synMD5( const Value : AnsiString ) : AnsiString');
14761: Function HMAC_MD5( Text, Key : AnsiString ) : AnsiString');
14762: Function MD5LongHash( const Value : AnsiString; Len : integer ) : AnsiString');
14763: Function synSHA1( const Value : AnsiString ) : AnsiString');
14764: Function HMAC_SHA1( Text, Key : AnsiString ) : AnsiString');
14765: Function SHA1LongHash( const Value : AnsiString; Len : integer ) : AnsiString');
14766: Function synMD4( const Value : AnsiString ) : AnsiString');
14767: end;
14768:
14769: procedure SIRegister_synamachar(CL: TPSPPascalCompiler);
14770: begin
14771:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14772:             + 'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14773:             + 'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14774:             + '4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14775:             + '_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE, '
14776:             + 'C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14777:             + ', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14778:             + ', NEXTSTEP, ARMSCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14779:             + 'IS620, CP874, VISCN, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14780:             + '1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14781:             + '2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14782:             + 'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14783:             + 'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14784:             + ', CP864, CP865, CP869, CP1125 )');
14785:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14786:   Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14787:   Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
14788:     TransformTable : array of Word) : AnsiString );
14789:   Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
14790:     TransformTable : array of Word; Translit : Boolean) : AnsiString );
14791:   Function GetCurCP : TMimeChar');
14792:   Function GetCurOEMCP : TMimeChar');
14793:   Function GetCFFromID( Value : AnsiString ) : TMimeChar');
14794:   Function GetIDFromCP( Value : TMimeChar ) : AnsiString );
14795:   Function NeedCharsetConversion( const Value : AnsiString ) : Boolean );
14796:   Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14797:   Function GetBOM( Value : TMimeChar ) : AnsiString );
14798:   Function StringToWide( const Value : AnsiString ) : WideString );
14799:   Function WideToString( const Value : WideString ) : AnsiString );
14800: end;
14801: procedure SIRegister_synamisc(CL: TPSPPascalCompiler);
14802: begin
14803:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14804:   Procedure WakeOnLan( MAC, IP : string );
14805:   Function GetDNS : string );
14806:   Function GetIEProxy( protocol : string ) : TProxySetting );
14807:   Function GetLocalIPs : string );
14808: end;
14809:
14810: procedure SIRegister_synaser(CL: TPSPPascalCompiler);
14811: begin
14812:   AddConstantN('synCR','Char #$0d');
14813:   Const('synLF','Char #$0a');
14814:   Const('cSerialChunk','LongInt'( 8192));
14815:   Const('LockfileDirectory','String '/var/lock');
14816:   Const('PortIsClosed','LongInt'( - 1);
14817:   Const('ErrAlreadyOwned','LongInt'( 9991);
14818:   Const('ErrAlreadyInUse','LongInt'( 9992);
14819:   Const('ErrWrongParameter','LongInt'( 9993);
14820:   Const('ErrPortNotOpen','LongInt'( 9994);
14821:   Const('ErrNoDeviceAnswer','LongInt'( 9995);
14822:   Const('ErrMaxBuffer','LongInt'( 9996);
14823:   Const('ErrTimeout','LongInt'( 9997);
14824:   Const('ErrNotRead','LongInt'( 9998);
14825:   Const('ErrFrame','LongInt'( 9999);
14826:   Const('ErrOverrun','LongInt'( 10000);
14827:   Const('ErrRxOver','LongInt'( 10001);
14828:   Const('ErrRxParity','LongInt'( 10002);
14829:   Const('ErrTxFull','LongInt'( 10003);
14830:   Const('dcb_Binary','LongWord')( $00000001);

```

```

14831: Const('dcb_ParityCheck','LongWord')( $00000002);
14832: Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14833: Const('dcb_OutxCsrFlow','LongWord')( $00000008);
14834: Const('dcb_DtrControlMask','LongWord')( $00000030);
14835: Const('dcb_DtrControlDisable','LongWord')( $00000000);
14836: Const('dcb_DtrControlEnable','LongWord')( $00000010);
14837: Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14838: Const('dcb_DsrSensitivity','LongWord')( $00000040);
14839: Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14840: Const('dcb_OutX','LongWord')( $00000100);
14841: Const('dcb_InX','LongWord')( $00000200);
14842: Const('dcb_ErrorChar','LongWord')( $00000400);
14843: Const('dcb_NullStrip','LongWord')( $00000800);
14844: Const('dcb_RtsControlMask','LongWord')( $00003000);
14845: Const('dcb_RtsControlDisable','LongWord')( $00000000);
14846: Const('dcb_RtsControlEnable','LongWord')( $00001000);
14847: Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14848: Const('dcb_RtsControlToggle','LongWord')( $00003000);
14849: Const('dcb_AbortOnError','LongWord')( $00004000);
14850: Const('dcb_Reservesd','LongWord')( $FFFF8000);
14851: Const('synSBL','LongInt'( 0));
14852: Const('SBlandHalf','LongInt'( 1));
14853: Const('synSB2','LongInt'( 2));
14854: Const('synINVALID_HANDLE_VALUE','LongInt'( THandle( - 1 )));
14855: Const('CS7fix','LongWord')( $0000020);
14856: AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long' +
14857:   + 'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par' +
14858:   + 'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : ' +
14859:   + 'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14860: //AddTypeS('PDCB', '^TDCB // will not work');
14861: //Const('MaxRates','LongInt'( 18));
14862: //Const('MaxRates','LongInt'( 30));
14863: //Const('MaxRates','LongInt'( 19));
14864: Const('O_SYNC','LongWord')( $0080);
14865: Const('synOK','LongInt'( 0));
14866: Const('synErr','LongInt'( integer( - 1 )));
14867: AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,' +
14868:   HR_WriteCount, HR_Wait )');
14869: Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string'));
14870: SIRegister_ESynaSerError(CL);
14871: SIRegister_TBlockSerial(CL);
14872: Function GetSerialPortNames : string);
14873: end;
14874: procedure SIRegister_synaicnv(CL: TPPSPascalCompiler);
14875: begin
14876: Const('DLLIconvName','String 'libiconv.so');
14877: Const('DLLIconvName','String 'iconv.dll');
14878: AddTypeS('size_t','Cardinal');
14879: AddTypeS('iconv_t','Integer');
14880: //AddTypeS('iconv_t','Pointer');
14881: AddTypeS('argptr','iconv_t');
14882: Function SynalconvOpen( const tocode, fromcode : Ansistring) : iconv_t';
14883: Function SynalconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t';
14884: Function SynalconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t';
14885: Function Synalconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer';
14886: Function SynalconvClose( var cd : iconv_t) : integer';
14887: Function SynalconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer';
14888: Function IsIconvloaded : Boolean';
14889: Function InitIconvInterface : Boolean';
14890: Function DestroyIconvInterface : Boolean';
14891: Const('ICONV_TRIVIALP','LongInt'( 0));
14892: Const('ICONV_GET_TRANSLITERATE','LongInt'( 1));
14893: Const('ICONV_SET_TRANSLITERATE','LongInt'( 2));
14894: Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3));
14895: Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4));
14896: end;
14897:
14898: procedure SIRegister_pingsend(CL: TPPSPascalCompiler);
14899: begin
14900: Const 'ICMP_ECHO','LongInt'( 8);
14901: Const('ICMP_ECHOREPLY','LongInt'( 0);
14902: Const('ICMP_UNREACH','LongInt'( 3);
14903: Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14904: Const('ICMP6_ECHO','LongInt'( 128);
14905: Const('ICMP6_ECHOREPLY','LongInt'( 129);
14906: Const('ICMP6_UNREACH','LongInt'( 1);
14907: Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14908: AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOr' +
14909:   + 'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14910: SIRegister_TPINGSend(CL);
14911: Function PingHost( const Host : string) : Integer';
14912: Function TraceRouteHost( const Host : string) : string';
14913: end;
14914:
14915: procedure SIRegister_asn1util(CL: TPPSPascalCompiler);
14916: begin
14917: AddConstantN('synASN1_BOOL','LongWord')( $01);
14918: Const('synASN1_INT','LongWord')( $02);

```

```

14919: Const('synASN1_OCTSTR','LongWord')( $04);
14920: Const('synASN1_NULL','LongWord')( $05);
14921: Const('synASN1_OBJID','LongWord')( $06);
14922: Const('synASN1_ENUM','LongWord')( $0a);
14923: Const('synASN1_SEQ','LongWord')( $30);
14924: Const('synASN1_SETOF','LongWord')( $31);
14925: Const('synASN1_IPADDR','LongWord')( $40);
14926: Const('synASN1_COUNTER','LongWord')( $41);
14927: Const('synASN1_GAUGE','LongWord')( $42);
14928: Const('synASN1_TIMETICKS','LongWord')( $43);
14929: Const('synASN1_OPAQUE','LongWord')( $44);
14930: Function synASNEncOIDItem( Value : Integer ) : AnsiString';
14931: Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString ) : Integer');
14932: Function synASNEncLen( Len : Integer ) : AnsiString';
14933: Function synASNDecLen( var Start : Integer; const Buffer : AnsiString ) : Integer');
14934: Function synASNEncInt( Value : Integer ) : AnsiString';
14935: Function synASNEncUInt( Value : Integer ) : AnsiString';
14936: Function synASNObject( const Data : AnsiString; ASNType : Integer ) : AnsiString';
14937: Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14938: Function synMibToId( Mib : String ) : AnsiString';
14939: Function synIdToMib( const Id : AnsiString ) : String';
14940: Function synIntMibToStr( const Value : AnsiString ) : AnsiString';
14941: Function ASNdump( const Value : AnsiString ) : AnsiString';
14942: Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings ) : Boolean';
14943: Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString';
14944: end;
14945:
14946: procedure SIRegister_ldapsend(CL: TPSCompiler);
14947: begin
14948: Const('cLDAPProtocol','String '389');
14949: Const('LDAP ASN1 BIND REQUEST','LongWord')( $60);
14950: Const('LDAP ASN1 BIND RESPONSE','LongWord')( $61);
14951: Const('LDAP ASN1 UNBIND REQUEST','LongWord')( $42);
14952: Const('LDAP ASN1 SEARCH REQUEST','LongWord')( $63);
14953: Const('LDAP ASN1 SEARCH ENTRY','LongWord')( $64);
14954: Const('LDAP ASN1 SEARCH DONE','LongWord')( $65);
14955: Const('LDAP ASN1 SEARCH REFERENCE','LongWord')( $73);
14956: Const('LDAP ASN1 MODIFY REQUEST','LongWord')( $66);
14957: Const('LDAP ASN1 MODIFY RESPONSE','LongWord')( $67);
14958: Const('LDAP ASN1 ADD REQUEST','LongWord')( $68);
14959: Const('LDAP ASN1 ADD RESPONSE','LongWord')( $69);
14960: Const('LDAP ASN1 DEL REQUEST','LongWord')( $4A);
14961: Const('LDAP ASN1 DEL RESPONSE','LongWord')( $6B);
14962: Const('LDAP ASN1 MODIFYDN REQUEST','LongWord')( $6C);
14963: Const('LDAP ASN1 MODIFYDN RESPONSE','LongWord')( $6D);
14964: Const('LDAP ASN1 COMPARE REQUEST','LongWord')( $5E);
14965: Const('LDAP ASN1 COMPARE RESPONSE','LongWord')( $6F);
14966: Const('LDAP ASN1 ABANDON REQUEST','LongWord')( $70);
14967: Const('LDAP ASN1 EXT REQUEST','LongWord')( $77);
14968: Const('LDAP ASN1 EXT RESPONSE','LongWord')( $78);
14969: SIRegister_TLDAPAttribute(CL);
14970: SIRegister_TLDAPAttributeList(CL);
14971: SIRegister_TLDAPResult(CL);
14972: SIRegister_TLDAPResultList(CL);
14973: AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14974: AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14975: AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14976: SIRegister_TLDAPSnd(CL);
14977: Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString';
14978: end;
14979:
14980:
14981: procedure SIRegister_slogsend(CL: TPSCompiler);
14982: begin
14983: Const('cSysLogProtocol','String '514');
14984: Const('FCL_Kernel','LongInt'( 0));
14985: Const('FCL_UserLevel','LongInt'( 1));
14986: Const('FCL_MailSystem','LongInt'( 2));
14987: Const('FCL_System','LongInt'( 3));
14988: Const('FCL_Security','LongInt'( 4));
14989: Const('FCL_Syslogd','LongInt'( 5));
14990: Const('FCL_Printer','LongInt'( 6));
14991: Const('FCL_News','LongInt'( 7));
14992: Const('FCL_UUCP','LongInt'( 8));
14993: Const('FCL_Clock','LongInt'( 9));
14994: Const('FCL_Authorization','LongInt'( 10));
14995: Const('FCL_FTP','LongInt'( 11));
14996: Const('FCL_NTP','LongInt'( 12));
14997: Const('FCL_LogAudit','LongInt'( 13));
14998: Const('FCL_LogAlert','LongInt'( 14));
14999: Const('FCL_Time','LongInt'( 15));
15000: Const('FCL_Local0','LongInt'( 16));
15001: Const('FCL_Local1','LongInt'( 17));
15002: Const('FCL_Local2','LongInt'( 18));
15003: Const('FCL_Local3','LongInt'( 19));
15004: Const('FCL_Local4','LongInt'( 20));
15005: Const('FCL_Local5','LongInt'( 21));
15006: Const('FCL_Local6','LongInt'( 22));
15007: Const('FCL_Local7','LongInt'( 23));

```

```

15008: Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
15009:   SIRegister_TSyslogMessage(CL);
15010:   SIRegister_TSyslogSend(CL);
15011:   Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
Content:string):Boolean;
15012: end;
15013:
15014:
15015: procedure SIRegister_mimemess(CL: TPSCompiler);
15016: begin
15017:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
15018:   SIRegister_TMessHeader(CL);
15019:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
15020:   SIRegister_TMimeMess(CL);
15021: end;
15022:
15023: procedure SIRegister_mimepart(CL: TPSCompiler);
15024: begin
15025:   (FindClass('TOBJECT'),'TMimePart');
15026:   AddTypes('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
15027:   AddTypes('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
15028:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
15029:   SIRegister_TMimePart(CL);
15030:   Const('MaxMineType','LongInt'( 25));
15031:   Function GenerateBoundary : string';
15032: end;
15033:
15034: procedure SIRegister_mimeinln(CL: TPSCompiler);
15035: begin
15036:   Function InlineDecode( const Value : string; CP : TMimeChar) : string';
15037:   Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string';
15038:   Function NeedInline( const Value : AnsiString) : boolean';
15039:   Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
15040:   Function InlineCode( const Value : string) : string');
15041:   Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
15042:   Function InlineEmail( const Value : string) : string');
15043: end;
15044:
15045: procedure SIRegister_ftpsend(CL: TPSCompiler);
15046: begin
15047:   Const('cFtpProtocol','String '21');
15048:   Const('cFtpDataProtocol','String '20');
15049:   Const('FTP_OK','LongInt'( 255);
15050:   Const('FTP_ERR','LongInt'( 254);
15051:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string )');
15052:   SIRegister_TFTPLListRec(CL);
15053:   SIRegister_TFTPLList(CL);
15054:   SIRegister_TFTPSend(CL);
15055:   Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean';
15056:   Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean';
15057:   Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
ToPort, ToFile, ToUser, ToPass : string) : Boolean';
15058: end;
15059:
15060: procedure SIRegister_httpsend(CL: TPSCompiler);
15061: begin
15062:   Const('cHttpProtocol','String '80');
15063:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15064:   SIRegister_THTTFSend(CL);
15065:   Function HttpGetText( const URL : string; const Response : TStrings) : Boolean';
15066:   Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean';
15067:   Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean';
15068:   Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean';
15069:   Function HttpPostfile(const URL,FieldName,FileName:string;const Data:TStream;const
ResultData:TStrings):Bool;
15070: end;
15071:
15072: procedure SIRegister_smtpsend(CL: TPSCompiler);
15073: begin
15074:   Const('cSmtpProtocol','String '25');
15075:   SIRegister_TSMTSPSend(CL);
15076:   Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
Passw:string):Bool;
15077:   Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15078:   Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
Usrname, Password : string):Boolean';
15079: end;
15080:
15081: procedure SIRegister_snmpsend(CL: TPSCompiler);
15082: begin
15083:   Const('cSnmpProtocol','String '161');
15084:   Const('cSnmpTrapProtocol','String '162');
15085:   Const('SNMP_V1','LongInt'( 0);
15086:   Const('SNMP_V2C','LongInt'( 1);
15087:   Const('SNMP_V3','LongInt'( 3);
15088:   Const('PDUGetRequest','LongWord')($A0);
15089:   Const('PDUGetNextRequest','LongWord')($A1);
15090:   Const('PDUGetResponse','LongWord')($A2);
15091:   Const('PDUSetRequest','LongWord')($A3);

```

```

15092: Const('PDUTrap','LongWord')($A4);
15093: Const('PDUGetBulkRequest','LongWord')($A5);
15094: Const('PDUInformRequest','LongWord')($A6);
15095: Const('PDUTrapV2','LongWord')($A7);
15096: Const('PDUReport','LongWord')($A8);
15097: Const('ENoError',LongInt 0;
15098: Const('ETooBig','LongInt')( 1);
15099: Const('ENoSuchName','LongInt'( 2);
15100: Const('EBadValue','LongInt'( 3);
15101: Const('EReadOnly','LongInt'( 4);
15102: Const('EGenErr','LongInt'( 5);
15103: Const('ENOAccess','LongInt'( 6);
15104: Const('EWrongType','LongInt'( 7);
15105: Const('EWrongLength','LongInt'( 8);
15106: Const('EWrongEncoding','LongInt'( 9);
15107: Const('EWrongValue','LongInt'( 10);
15108: Const('ENOCreation','LongInt'( 11);
15109: Const('EInconsistentValue','LongInt'( 12);
15110: Const('EResourceUnavailable','LongInt'( 13);
15111: Const('ECommitFailed','LongInt'( 14);
15112: Const('EUndoFailed','LongInt'( 15);
15113: Const('EAuthorizationError','LongInt'( 16);
15114: Const('ENotWritable','LongInt'( 17);
15115: Const('EInconsistentName','LongInt'( 18);
15116: AddTypes('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15117: AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15118: AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15119: SIRegister_TSNSMPMib(CL);
15120: AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer;
15121: +EngineTime : integer; EngineStamp : Cardinal; end');
15122: SIRegister_TSNSMPRec(CL);
15123: SIRegister_TSNSMPSend(CL);
15124: Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString ) : Boolean';
15125: Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer ) : Boolean';
15126: Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15127: Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings) : Boolean;
15128: Function SNMPGetTableElement(const BaseOID,RowID,Colid,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15129: Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds : Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer ) : Integer';
15130: Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds : Integer; const MIBName, MIBValue : TStringList ) : Integer';
15131: end;
15132:
15133: procedure SIRegister_NetWork(CL: TPPascalCompiler);
15134: begin
15135:   Function GetDomainName2: AnsiString';
15136:   Function GetDomainController( Domain : AnsiString ) : AnsiString';
15137:   Function GetDomainUsers( Controller : AnsiString ) : AnsiString';
15138:   Function GetDomainGroups( Controller : AnsiString ) : AnsiString';
15139:   Function GetDateTime( Controller : AnsiString ) : TDateTime';
15140:   Function GetFullName2( Controller, UserID : AnsiString ) : AnsiString';
15141: end;
15142:
15143: procedure SIRegister_wwSystem(CL: TPPascalCompiler);
15144: begin
15145:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15146:   TwwDateTimeSelection', '(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM)';
15147:   Function wwStrToDate( const S : string ) : boolean';
15148:   Function wwStrToTime( const S : string ) : boolean';
15149:   Function wwStrToDateTime( const S : string ) : boolean';
15150:   Function wwStrToTimeVal( const S : string ) : TDateTime';
15151:   Function wwStrToDateVal( const S : string ) : TDateTime';
15152:   Function wwStrToDateTimeVal( const S : string ) : TDateTime';
15153:   Function wwStrToInt( const S : string ) : boolean';
15154:   Function wwStrToFloat( const S : string ) : boolean';
15155:   Function wwGetDateOrder( const DateFormat : string ) : TwwDateOrder';
15156:   Function wwNextDay( Year, Month, Day : Word ) : integer';
15157:   Function wwPriorDay( Year, Month, Day : Word ) : integer';
15158:   Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime ) : Boolean';
15159:   Function wwDoDecodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime ) : Boolean';
15160:   Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool:TwwDateTimeSelection');
15161:   Function wwGetTimeCursorPosition( SelStart : integer; Text : string ) : TwwDateTimeSelection';
15162:   Function wwScanDate( const S : string; var Date : TDateTime ) : Boolean';
15163:   Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer ) : Boolean';
15164:   Procedure wwSetDateTimeCursorPosition(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15165:   Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean';
15166: end;
15167:
15168: unit uPSI_Themes;
15169: Function ThemeServices : TThemeServices';
15170: Function ThemeControl( AControl : TControl ) : Boolean';
15171: Function UnthemedDesigner( AControl : TControl ) : Boolean';
15172: procedure SIRegister_UDDIHelper(CL: TPPascalCompiler);
15173: begin
15174:   Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String';
15175: end;
15176: Unit upSC_menus;
15177:   Function StripHotkey( const Text : string ) : string';
15178:   Function GetHotkey( const Text : string ) : string';

```

```

15179: Function AnsiSameCaption( const Text1, Text2 : string ) : Boolean';
15180: Function IsAltGRRPressed : boolean';
15181:
15182: procedure SIRегистер_IdIMAP4Server(CL: TPSPascalCompiler);
15183: begin
15184:   TCommandEvent', 'Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean)';
15185:   SIRегистер_TIdIMAP4Server(CL);
15186: end;
15187:
15188: procedure SIRегистер_VariantSymbolTable(CL: TPSPascalCompiler);
15189: begin
15190:   'HASH_SIZE', 'LongInt'( 256 );
15191:   CL.FindClass('TOBJECT'), 'EVariantSymbolTable');
15192:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15193:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15194:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15195:     + ' : Integer; Value : Variant; end');
15196:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15197:   SIRегистер_TVariantSymbolTable(CL);
15198: end;
15199:
15200: procedure SIRегистер_udf_glob(CL: TPSPascalCompiler);
15201: begin
15202:   SIRегистер_ThreadLocalVariables(CL);
15203:   Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD ) : PChar';
15204:   //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD ) : PISC_QUAD';
15205:   Function ThreadLocals : TThreadLocalVariables';
15206:   Procedure WriteDebug( sz : String );
15207:   CL.AddConstantN('UDF_SUCCESS', 'LongInt'( 0 );
15208:   'UDF_FAILURE', 'LongInt'( 1 );
15209:   'cSignificantlyLarger', 'LongInt'( 1024 * 4 );
15210:   CL.AddTypeS('mTByteArray', 'array of byte;');
15211:   function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15212:   function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15213:   procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTTarget: string);
15214:   function IsNetworkConnected: Boolean;
15215:   function IsInternetConnected: Boolean;
15216:   function IsCOMConnected: Boolean;
15217:   function IsNetworkOn: Boolean;
15218:   function IsInternetOn: Boolean;
15219:   function IsCOMON: Boolean;
15220:   Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15221:   TmrProc', 'procedure TmrProc(hWnd: HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);');
15222:   Function SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc ) : UINT';
15223:   Function KillTimer( hWnd : HWND; uIDEvent : UINT ) : BOOL';
15224:   Function wIsWindowUnicode( hWnd : HWND ) : BOOL';
15225:   Function wEnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL';
15226:   Function wIsWindowEnabled( hWnd : HWND ) : BOOL';
15227:   Function GetMenu( hWnd : HWND ) : HMENU';
15228:   Function SetMenu( hWnd : HWND; hMenu : HMENU ) : BOOL';
15229: end;
15230:
15231: procedure SIRегистер_SockTransport(CL: TPSPascalCompiler);
15232: begin
15233:   SIRегистер_IDataBlock(CL);
15234:   SIRегистер_ISendDataBlock(CL);
15235:   SIRегистер_ITransport(CL);
15236:   SIRегистер_TDataBlock(CL);
15237:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15238:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15239:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15240:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15241:   SIRегистер_TCustomDataBlockInterpreter(CL);
15242:   SIRегистер_TSendDataBlock(CL);
15243:   'CallSig', 'LongWord')($D800);
15244:   'ResultsSig', 'LongWord')($D400);
15245:   'asMask', 'LongWord')($00FF);
15246:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EInterpreterError');
15247:   CL.AddClassN(CL.FindClass('TOBJECT'), 'ESocketConnectionError');
15248:   Procedure CheckSignature( Sig : Integer );
15249: end;
15250:
15251: procedure SIRегистер_WinInet(CL: TPSPascalCompiler);
15252: begin
15253:   //CL.AddTypeS('HINTERNET', '__Pointer');
15254:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15255:   CL.AddTypeS('HINTERNET', 'Integer');
15256:   CL.AddTypeS('HINTERNET2', '__Pointer');
15257:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15258:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15259:   CL.AddTypeS('INTERNET_PORT', 'Word');
15260:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15261:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15262:   Function InternetTimeFromSystemTime( const pst: TSystemTime; dwRFC: DWORD; lpszTime: LPSTR; cbTime: DWORD ) : BOOL;
15263:   'INTERNET_RFC1123_FORMAT', 'LongInt'( 0 );
15264:   'INTERNET_RFC1123_BUFSIZE', 'LongInt'( 30 );
15265:   Function InternetCrackUrl(lpszUrl: PChar; dwUrlLength, dwFlags: DWORD; var
15266:   lpUrlComponents: TURLComponents) : BOOL;
15267:   Function InternetCreateUrl(var lpUrlComponents: TURLComponents; dwFlags: DWORD; lpszUrl: PChar; var
15268:   dwUrlLength: DWORD) : BOOL;

```

```

15267: Function InternetCloseHandle(hInet : HINTERNET) : BOOL';
15268: Function
InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15269: Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15270: ;dwContext:DWORD):HINTERNET;
15271: Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15272: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
dwContext:DWORD):BOOL;
15273: Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE ) : BOOL';
15274: Function
InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15275: Function InternetHangUp(dwConnection : DWORD; dwReserved : DWORD) : DWORD';
15276: Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD ) : BOOL';
15277: Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15278: Function InternetAutodialHangup( dwReserved : DWORD ) : BOOL';
15279: Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15280: Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15281: Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET ) : BOOL');
15282: Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15283: ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15284: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
dwContext:DWORD):BOOL;
15285: Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
TWin32FindData; dwFlags : DWORD; dwContext : DWORD ) : HINTERNET';
15286: Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD ) : BOOL';
15287: Function
WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15288: Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar ) : BOOL';
15289: Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15290: Function
FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15291: Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar ) : BOOL';
15292: Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar ) : BOOL';
15293: Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar ) : BOOL';
15294: Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15295: Function
FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15296: Function IS_GOPHER_FILE( GopherType : DWORD ) : BOOL';
15297: Function IS_GOPHER_DIRECTORY( GopherType : DWORD ) : BOOL';
15298: Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD ) : BOOL';
15299: Function IS_GOPHER_ERROR( GopherType : DWORD ) : BOOL';
15300: Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD ) : BOOL';
15301: Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD ) : BOOL';
15302: Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD ) : BOOL';
15303: Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD ) : BOOL';
15304: Function IS_GOPHER_ASK( GopherType : DWORD ) : BOOL';
15305: Function IS_GOPHER_PLUS( GopherType : DWORD ) : BOOL';
15306: Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD ) : BOOL';
15307: Function
GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
PChar;dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15308: Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL';
15309: Function
GopherOpenFile(hConct:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15310: Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15311: Function
HttpAddRequestHeaders(hReg:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15312: Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15313: Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15314: Function InternetAttemptConnect( dwReserved : DWORD ) : DWORD';
15315: Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15316: Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15317: Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LESTR; bPost : BOOL ) : DWORD';
15318: Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject ) : Int64';
15319: Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject ) : Bool';
15320: Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15321: Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15322: Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15323: Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15324: Function
InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15325: Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD ) : BOOL';
15326: end;
15327:
15328: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15329: begin
15330:   AddTypeS('strCharSet', 'set of char');
15331:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords );
15332:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15333:   Procedure strBreakApart( s : string; delimiter : string; parts : TStrings)');

```

```

15334: Function strGetToken( s : string; delimiter : string; var APos : integer ) : string');
15335: Procedure strStripPreceding( var s : string; delimiter : strCharSet );
15336: Procedure strStripTrailing( var s : string; delimiter : strCharSet );
15337: Procedure strStripWhiteSpace( var s : string );
15338: Function strRemoveChar( str : string; removeChar : char ) : string';
15339: Function wwstrReplaceChar( str : string; removeChar, replaceChar : char ) : string');
15340: Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string ) : string');
15341: Function wwEqualStr( s1, s2 : string ) : boolean';
15342: Function strCount( s : string; delimiter : char ) : integer';
15343: Function strWhiteSpace : strCharSet );
15344: Function wwExtractFileNameOnly( const FileName : string ) : string');
15345: Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15346: Function strTrailing( s : string; delimiter : char ) : string');
15347: Function strPreceding( s : string; delimiter : char ) : string');
15348: Function wwstrReplace( s, Find, Replace : string ) : string');
15349: end;
15350:
15351: procedure SIRegister_DataBkr(CL: TPSPPascalCompiler);
15352: begin
15353:   SIRegister_TRemoteDataModule(CL);
15354:   SIRegister_TCRemoteDataModule(CL);
15355:   Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean );
15356:   Procedure UnregisterPooled( const ClassID : string );
15357:   Procedure EnableSocketTransport( const ClassID : string );
15358:   Procedure DisableSocketTransport( const ClassID : string );
15359:   Procedure EnableWebTransport( const ClassID : string );
15360:   Procedure DisableWebTransport( const ClassID : string );
15361: end;
15362:
15363: procedure SIRegister_Mathbox(CL: TPSPPascalCompiler);
15364: begin
15365:   Function mxArcCos( x : Real ) : Real';
15366:   Function mxArcSin( x : Real ) : Real';
15367:   Function Comp2Str( N : Comp ) : String';
15368:   Function Int2StrPad0( N : LongInt; Len : Integer ) : String';
15369:   Function Int2Str( N : LongInt ) : String';
15370:   Function mxIsEqual( R1, R2 : Double ) : Boolean';
15371:   Function LogXY( x, y : Real ) : Real';
15372:   Function Pennies2Dollars( C : Comp ) : String';
15373:   Function mxPower( X : Integer; Y : Integer ) : Real';
15374:   Function Real2Str( N : Real; Width, Places : integer ) : String';
15375:   Function mxStr2Comp( MyString : string ) : Comp';
15376:   Function mxStr2Pennies( S : String ) : Comp';
15377:   Function Str2Real( MyString : string ) : Real';
15378:   Function XToTheY( x, y : Real ) : Real';
15379: end;
15380:
15381: //*****Cindy Functions!*****
15382: procedure SIRegister_cyIndy(CL: TPSPPascalCompiler);
15383: begin
15384:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15385:     + '_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15386:     + 'cmAlterText_Html_RelatedAttach, cmAlterText_Html_Attach_RelatedAttach, cmReadNotification )');
15387:   MessagePlainText', 'String 'text/plain');
15388:   CL.AddConstantN('MessagePlainText_Attach', 'String 'multipart/mixed');
15389:   MessageAlterText_Html', 'string 'multipart/alternative');
15390:   MessageHtml_Attach', 'String 'multipart/mixed');
15391:   MessageHtml_RelatedAttach', 'String 'multipart/related; type="text/html"');
15392:   MessageAlterText_Html_Attach', 'String 'multipart/mixed');
15393:   MessageAlterText_Html_RelatedAttach', 'String ')('multipart/related;type="multipart/alternative"');
15394:   MessageAlterText_Html_Attach_RelatedAttach', 'String 'multipart/mixed');
15395:   MessageReadNotification', 'String ')(('multipart/report; report-type="disposition-notification" ';
15396:   Function ForceDecodeHeader( aHeader : String ) : String');
15397:   Function Base64_EncodeString( Value : String; const aEncoding : TEncoding ) : string');
15398:   Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding ) : String');
15399:   Function Base64_DecodeToBytes( Value : String ) : TBytes;');
15400:   Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15401:   Function Get_MD5( const aFileName : string ) : string');
15402:   Function Get_MD5FromString( const aString : string ) : string');
15403: end;
15404:
15405: procedure SIRegister_cySysUtils(CL: TPSPPascalCompiler);
15406: begin
15407:   Function IsFolder( SRec : TSearchrec ) : Boolean';
15408:   Function isFolderReadOnly( Directory : String ) : Boolean';
15409:   Function DirectoryIsEmpty( Directory : String ) : Boolean';
15410:   Function DirectoryWithSubDir( Directory : String ) : Boolean';
15411:   Procedure GetSubDirs( FromDirectory : String; aList : TStrings );
15412:   Function DiskFreeBytes( Drv : Char ) : Int64';
15413:   Function DiskBytes( Drv : Char ) : Int64';
15414:   Function GetFileBytes( Filename : String ) : Int64';
15415:   Function GetFilesBytes( Directory, Filter : String ) : Int64';
15416:   SE_CREATE_TOKEN_NAME', 'String 'SeCreateTokenPrivilege');
15417:   SE_ASSIGNPRIMARYTOKEN_NAME', 'String 'SeAssignPrimaryTokenPrivilege');
15418:   SE_LOCK_MEMORY_NAME', 'String 'SeLockMemoryPrivilege');
15419:   SE_INCREASE_QUOTA_NAME', 'String 'SeIncreaseQuotaPrivilege');
15420:   SE_UNSOLICITED_INPUT_NAME', 'String 'SeUnsolicitedInputPrivilege');
15421:   SE_MACHINE_ACCOUNT_NAME', 'String 'SeMachineAccountPrivilege');
15422:   SE_TCB_NAME', 'String 'SeTcbPrivilege');

```

```

15423: SE_SECURITY_NAME', 'String 'SeSecurityPrivilege');
15424: SE_TAKE_OWNERSHIP_NAME', 'String 'SeTakeOwnershipPrivilege');
15425: SE_LOAD_DRIVER_NAME', 'String 'SeLoadDriverPrivilege');
15426: SE_SYSTEM_PROFILE_NAME', 'String 'SeSystemProfilePrivilege');
15427: SE_SYSTEMTIME_NAME', 'String 'SeSystemtimePrivilege');
15428: SE_PROF_SINGLE_PROCESS_NAME', 'String 'SeProfileSingleProcessPrivilege');
15429: SE_INC_BASE_PRIORITY_NAME', 'String 'SeIncreaseBasePriorityPrivilege');
15430: SE_CREATE_PAGEFILE_NAME', 'String 'SeCreatePagefilePrivilege');
15431: SE_CREATE_PERMANENT_NAME', 'String 'SeCreatePermanentPrivilege');
15432: SE_BACKUP_NAME', 'String 'SeBackupPrivilege');
15433: SE_RESTORE_NAME', 'String 'SeRestorePrivilege');
15434: SE_SHUTDOWN_NAME', 'String 'SeShutdownPrivilege');
15435: SE_DEBUG_NAME', 'String 'SeDebugPrivilege');
15436: SE_AUDIT_NAME', 'String 'SeAuditPrivilege');
15437: SE_SYSTEM_ENVIRONMENT_NAME', 'String 'SeSystemEnvironmentPrivilege');
15438: SE_CHANGE_NOTIFY_NAME', 'String 'SeChangeNotifyPrivilege');
15439: SE_REMOTE_SHUTDOWN_NAME', 'String 'SeRemoteShutdownPrivilege');
15440: SE_UNDOCK_NAME', 'String 'SeUndockPrivilege');
15441: SE_SYNC_AGENT_NAME', 'String 'SeSyncAgentPrivilege');
15442: SE_ENABLE_DELEGATION_NAME', 'String 'SeEnableDelegationPrivilege');
15443: SE_MANAGE_VOLUME_NAME', 'String 'SeManageVolumePrivilege');
15444: end;
15445:
15446: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15447: begin
15448:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15449:     + 'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15450:   Function ShellGetExtensionName( FileName : String ) : String';
15451:   Function ShellGetIconIndex( FileName : String ) : Integer';
15452:   Function ShellGetIconHandle( FileName : String ) : HIcon';
15453:   Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string );
15454:   Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string );
15455:   Procedure ShellRenameDir( DirFrom, DirTo : string );
15456:   Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15457:   Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15458:   Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String );
15459:   Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string );
15460:   Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
      WaitCloseCompletion : boolean ) : Boolean';
15461:   Procedure RestoreAndSetForegroundWindow( Hnd : Integer );
15462:   Function RemoveDuplicatedPathDelimiter( Str : String ) : String';
15463:   Function cyFileTimeToDate( _FT : TFileTime ) : TDateTime';
15464:   Function GetModificationDate( Filename : String ) : TDateTime';
15465:   Function GetCreationDate( Filename : String ) : TDateTime';
15466:   Function GetLastAccessDate( Filename : String ) : TDateTime';
15467:   Function FileDelete( Filename : String ) : Boolean';
15468:   Function FileIsOpen( Filename : string ) : boolean';
15469:   Procedure FileDelete( FromDirectory : String; Filter : ShortString );
15470:   Function DirectoryDelete( Directory : String ) : Boolean';
15471:   Function GetPrinters( PrintersList : TStrings ) : Integer';
15472:   Procedure SetDefaultPrinter( PrinterName : String );
15473:   Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer );
15474:   Function WinToDosPath( WinPathName : String ) : String';
15475:   Function DosToWinPath( DosPathName : String ) : String';
15476:   Function cyGetWindowsVersion : TWindowsVersion';
15477:   Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean ) : Boolean';
15478:   Procedure WindowsShutDown( Restart : boolean );
15479:   Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
      FileIcon:string;NumIcone:integer);
15480:   Procedure GetWindowsFonts( FontsList : TStrings );
15481:   Function GetAvailableFilename( DesiredFileName : String ) : String';
15482: end;
15483:
15484: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15485: begin
15486:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15487:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15488:   Type(TStringRead', '( srFromLeft, srFromRight )');
15489:   Type(TStringReads', 'set of TStringRead');
15490:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15491:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15492:   Type(TWordsOptions', 'set of TWordsOption');
15493:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15494:   Type(TCarTypes', 'set of TCarType');
15495:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15496:   CarTypeAlphabetic:LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15497:   Function Char_GetType( aChar : Char ) : TCarType';
15498:   Function SubString_Count( Str : String; Separator : Char ) : Integer';
15499:   Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15500:   Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word ) : String';
15501:   Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15502:   Procedure SubString_Add( var Str : String; Separator : Char; Value : String );
15503:   Procedure SubString_Insert(var Str:String;Separator:Char;SubStringIndex:Word;Value : String );
15504:   Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word;NewValue : String );
15505:   Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean';
15506:   Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15507:   Function SubString_Ribbon( Str : string; Separator : Char; Current : Word; MoveBy : Integer ):Integer';
15508:   Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String';
15509:   Function String_Quote( Str : String ) : String';

```

```

15510: Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char ) : Char';
15511: Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15512: Function String_GetWord( Str : String; StringRead : TStringRead ) : String');
15513: Function String_GetInteger( Str : String; StringRead : TStringRead ) : String');
15514: Function String_ToInt( Str : String ) : Integer');
15515: Function String_Uppercase( Str : String; Options : TWordsOptions ) : String');
15516: Function String_Lowercase( Str : String; Options : TWordsOptions ) : String');
15517: Function String_Reverse( Str : String ) : String');
15518: Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15519: Function String_Posl(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive : TCaseSensitive):Integer');
15520: Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer ) : String');
15521: Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String; Inclusive:Boolean):String;
15522: Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
Between2MustExist : Boolean; CaseSensitive : TCaseSensitive ) : String');
15523: Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer ) : String');
15524: Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String');
15525: Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads ) : String');
15526: Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer ) : String');
15527: Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String';
15528: Function String_End( Str : String; Cars : Word ) : String');
15529: Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
AlwaysFindFromBeginning : Boolean) : String');
15530: Function String_SubstCar( Str : String; Old, New : Char ) : String');
15531: Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive ) : Integer');
15532: Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15533: Function String_IsNumbers( Str : String ) : Boolean');
15534: Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer ) : Integer');
15535: Function StringToCsvCell( aStr : String ) : String');
15536: end;
15537:
15538: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15539: begin
15540:   Function LongDayName( aDate : TDate ) : String');
15541:   Function LongMonthName( aDate : TDate ) : String');
15542:   Function ShortYearOf( aDate : TDate ) : byte');
15543:   Function DateToStrYYYYMMDD( aDate : TDate ) : String');
15544:   Function StrYYYYMMDDToDate( aStr : String ) : TDate');
15545:   Function SecondsToMinutes( Seconds : Integer ) : Double');
15546:   Function MinutesToSeconds( Minutes : Double ) : Integer');
15547:   Function MinutesToHours( Minutes : Integer ) : Double');
15548:   Function HoursToMinutes( Hours : Double ) : Integer');
15549:   Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String ) : TDateTime');
15550:   Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer );
15551:   Function MergeDateWithTime( aDate : TDate; aTime : TDateTime ) : TDateTime');
15552:   Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime ) : Int64 );
15553:   Function GetMinutesBetween(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15554:   Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime ) : Int64 );
15555:   Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
RsltBegin:TDateTime; RsltEnd : TDateTime ) : Boolean );
15556:   Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15557:   Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime ) : Boolean );
15558: end;
15559:
15560: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15561: begin
15562:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended ) ');
15563:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended ) ');
15564:   Type(TcyLocateOption', '( lCaseInsensitive, 1PartialKey ) ');
15565:   Type(TcyLocateOptions', 'set of TcyLocateOption' );
15566:   Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions ) : Integer');
15567:   Function StringsLocatel( aList : TStrings; Value : String; ValueKind : TStringsValueKind ) : Integer');
15568:   Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15569:   Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind );
15570:   Procedure StringsSort( aList : TStrings; SortType : TStringsSortType );
15571:   Function TreeNodeLocate( ParentNode : TTTreeNode; Value : String ) : TTTreeNode');
15572:   Function TreeNodeLocateOnLevel( TreeView : TTTreeView; OnLevel : Integer; Value : String ) : TTTreeNode');
15573:   Function
TreeNodeGetChildFromIndex(TreeView:TTTreeView;ParentNode:TTTreeNode;ChildIndex:Integer):TTTreeNode';
15574:   Function TreeNodeGetParentOnLevel( ChildNode : TTTreeNode; ParentLevel : Integer ) : TTTreeNode');
15575:   Procedure TreeNodeCopy(FromNode:TTTreeNode;ToNode:TTTreeNode;const CopyChildren:Boolean;const
CopySubChildren:Bool;
15576:   Procedure RichEditSetStr( aRichEdit : TRichEdit; FormattedString : String );
15577:   Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15578:   Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint ) : TControl );
15579:   Procedure cyCenterControl( aControl : TControl );
15580:   Function GetLastParent( aControl : TControl ) : TWinControl );
15581:   Function GetControlBitmap( aControl : TWinControl ) : TBitmap );
15582:   Function GetRichEditBitmap( aRichEdit : TRichEdit ) : TBitmap );
15583: end;
15584:
15585: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15586: begin
15587:   Function TablePackTable( Tab : TTable ) : Boolean );
15588:   Function TableRegenIndexes( Tab : TTable ) : Boolean );
15589:   Function TableShowDeletedRecords( Tab : TTable; Show : Boolean ) : Boolean );
15590:   Function TableUndeleteRecord( Tab : TTable ) : Boolean );
15591:   Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15592:   Function TableDeleteIndex( Tab : TTable; IndexFieldName : String ) : Boolean );

```

```

15593: Function TableEmptyTable( Tab : TTable ) : Boolean';
15594: Function TableFindKey( aTable : TTable; Value : String ) : Boolean';
15595: Procedure TableFindNearest( aTable : TTable; Value : String );
15596: Function
  TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Bool):TTable;
15597: Function
  TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15598: Function DateToBDESQLODate( aDate : TDate; const DateFormat : String ) : String';
15599: end;
15600:
15601: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15602: begin
15603:   SIRegister_TcyRunTimeDesign(CL);
15604:   SIRegister_TcyShadowText(CL);
15605:   SIRegister_TcyBgPicture(CL);
15606:   SIRegister_TcyGradient(CL);
15607:   SIRegister_TcyBevel(CL);
15608:   //CL.AddTypes('TcyBevelClass', 'class of tcyBevel');
15609:   SIRegister_tcyBevels(CL);
15610:   SIRegister_TcyImagelistOptions(CL);
15611:   Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture );
15612: end;
15613:
15614: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15615: begin
15616:   Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
    adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
    Maxdegrade : Byte );
15617:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap );
15618:   Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15619:   Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15620:   Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
    Byte; toRect : TRect; OrientationShape : TDgradOrientationShape );
15621:   Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
    Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap );
15622:   Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer );
15623:   Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer );
15624:   Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
    BottomColor : TColor; const Width : Integer; const RoundRect : boolean );
15625:   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);
15626:   Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
    aState : TButtonState; Focused, Hot : Boolean );
15627:   Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
    aState : TButtonState; Focused, Hot : Boolean );
15628:   Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
    aState : TButtonState; Focused, Hot : Boolean );
15629:   Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
    GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean );
15630:   Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
    const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor );
15631:   Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
    TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer );
15632:   Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
    TALignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15633:   Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
    WordWrap : Boolean; CaptionRender : TCaptionRender ) : LongInt );
15634:   Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt );
15635:   Function cyCreateFontIndirect( fromFont : TFont; Angle : Double ) : TFont;
15636:   Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation ) : TFont;
15637:   Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
    CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout );
15638:   Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone : Integer;
    const OnlyCalcFoldLine : Boolean ) : TLineCoord;
15639:   Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone : Integer;
    const OnlyCalcFoldLine : Boolean ) : TLineCoord;
15640:   Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint ) : boolean;
15641:   Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint ) : boolean;
15642:   Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint ) : boolean;
15643:   Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint ) : boolean;
15644:   Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect );
15645:   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);
15646:   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 );
15647:   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 );
15648:   Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap );
15649:   Function ValidGraphic( aGraphic : TGraphic ) : Boolean;
15650:   Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer ) : TColor;
15651:   Function ColorModify( Color : TColor; incR, incG, incB : Integer ) : TColor;
15652:   Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer ) : TColor;
15653:   Function ColorSetPercentPale( Color : TColor; IncPercent : integer ) : TColor;

```

```

15654: Function MediumColor( Color1, Color2 : TColor ) : TColor');
15655: Function ClientToScreenRect( aControl : TControl; aControlRect : TRect ) : TRect');
15656: Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect ) : TRect');
15657: Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect ) : TRect');
15658: Procedure InflateRectPercent( var aRect : TRect; withPercent : Double );
15659: Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double ) : TRect');
15660: Function GetProportionalRect( fromRect, InsideRect : TRect ) : TRect');
15661: Function PointInRect( const aPt : TPoint; const aRect : TRect ) : boolean );
15662: Function PointInEllispe( const aPt : TPoint; const aRect : TRect ) : boolean );
15663: Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String ) : Integer );
15664: end;
15665:
15666: procedure SIRegister_cyTypes(CL: TPPSPascalCompiler);
15667: begin
15668:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15669:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15670:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15671:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15672:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15673:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,
15674:     +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15675:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15676:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15677:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15678:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15679:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
15680:     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: end;
15686: procedure SIRegister_WinSvc(CL: TPPSPascalCompiler);
15687: begin
15688:   Const SERVICES_ACTIVE_DATABASEA', 'String 'ServicesActive');
15689:   SERVICES_ACTIVE_DATABASEW', 'String') 'ServicesActive');
15690:   Const SERVICES_FAILED_DATABASEA', 'String') SERVICES_ACTIVE_DATABASEA');
15691:   Const SERVICES_FAILED_DATABASEW', 'String') 'ServicesFailed');
15692:   Const SERVICES_FAILED_DATABASE', 'String') SERVICES_FAILED_DATABASEA');
15693:   Const SC_GROUP_IDENTIFIERA', 'String'). '+');
15694:   Const SC_GROUP_IDENTIFIERW', 'String') '+');
15695:   Const SC_GROUP_IDENTIFIER', 'string 'SC_GROUP_IDENTIFIERA');
15696:   Const SERVICE_NO_CHANGE', 'LongWord $FFFFFF');
15697:   Const SERVICE_ACTIVE', 'LongWord')($00000001);
15698:   Const SERVICE_INACTIVE', 'LongWord $00000002);
15699:   Const SERVICE_CONTROL_STOP', 'LongWord $00000001);
15700:   Const SERVICE_CONTROL_PAUSE', 'LongWord $00000002);
15701:   Const SERVICE_CONTROL_CONTINUE', 'LongWord $00000003);
15702:   Const SERVICE_CONTROL_INTERROGATE', 'LongWord $00000004);
15703:   Const SERVICE_CONTROL_SHUTDOWN', 'LongWord $00000005);
15704:   Const SERVICE_STOPPED', 'LongWord $00000001);
15705:   Const SERVICE_START_PENDING', 'LongWord $00000002);
15706:   Const SERVICE_STOP_PENDING', 'LongWord $00000003);
15707:   Const SERVICE_RUNNING', 'LongWord $00000004);
15708:   Const SERVICE_CONTINUE_PENDING', 'LongWord $00000005);
15709:   Const SERVICE_PAUSE_PENDING', 'LongWord $00000006);
15710:   Const SERVICE_PAUSED', 'LongWord $00000007);
15711:   Const SERVICE_ACCEPT_STOP', 'LongWord $00000001);
15712:   Const SERVICE_ACCEPT_PAUSE_CONTINUE', 'LongWord $00000002);
15713:   Const SERVICE_ACCEPT_SHUTDOWN', 'LongWord $00000004);
15714:   Const SC_MANAGER_CONNECT', 'LongWord $0001);
15715:   Const SC_MANAGER_CREATE_SERVICE', 'LongWord $0002;
15716:   Const SC_MANAGER_ENUMERATE_SERVICE', 'LongWord $0004);
15717:   Const SC_MANAGER_LOCK', 'LongWord $0008);
15718:   Const SC_MANAGER_QUERY_LOCK_STATUS', 'LongWord $0010);
15719:   Const SC_MANAGER_MODIFY_BOOT_CONFIG', 'LongWord $0020);
15720:   Const SERVICE_QUERY_CONFIG', 'LongWord $0001);
15721:   Const SERVICE_CHANGE_CONFIG', 'LongWord $0002);
15722:   Const SERVICE_QUERY_STATUS', 'LongWord $0004);
15723:   Const SERVICE_ENUMERATE_DEPENDENTS', 'LongWord $0008);
15724:   Const SERVICE_START', 'LongWord $0010);
15725:   Const SERVICE_STOP', 'LongWord $0020);
15726:   Const SERVICE_PAUSE_CONTINUE', 'LongWord $0040);
15727:   Const SERVICE_INTERROGATE', 'LongWord $0080);
15728:   Const SERVICE_USER_DEFINED_CONTROL', 'LongWord $0100);
15729:   Const SERVICE_KERNEL_DRIVER', 'LongWord $00000001);
15730:   Const SERVICE_FILE_SYSTEM_DRIVER', 'LongWord $00000002);
15731:   Const SERVICE_ADAPTER', 'LongWord $00000004);
15732:   Const SERVICE_RECOGNIZER_DRIVER', 'LongWord $00000008);
15733:   Const SERVICE_WIN32_OWN_PROCESS', 'LongWord $00000010);
15734:   Const SERVICE_WIN32_SHARE_PROCESS', 'LongWord $00000020);
15735:   Const SERVICE_INTERACTIVE_PROCESS', 'LongWord $00000100);
15736:   Const SERVICE_BOOT_START', 'LongWord $00000000);
15737:   Const SERVICE_SYSTEM_START', 'LongWord $00000001);
15738:   Const SERVICE_AUTO_START', 'LongWord $00000002);
15739:   Const SERVICE_DEMAND_START', 'LongWord $00000003);
15740:   Const SERVICE_DISABLED', 'LongWord $00000004);

```

```

15741: Const SERVICE_ERROR_IGNORE', 'LongWord $00000000);
15742: Const SERVICE_ERROR_NORMAL', 'LongWord $00000001);
15743: Const SERVICE_ERROR_SEVERE', 'LongWord $00000002);
15744: Const SERVICE_ERROR_CRITICAL', 'LongWord $00000003);
15745: CL.AddTypeS('SC_HANDLE', 'THandle');
15746: //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15747: CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15748: Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15749: +' : DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15750: +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15751: Const SERVICE_STATUS', '_SERVICE_STATUS');
15752: Const TServiceStatus', '_SERVICE_STATUS');
15753: CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15754: +'playName : PChar; ServiceStatus : TServiceStatus; end');
15755: ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15756: _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15757: TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15758: TEnumServiceStatus', 'TEnumServiceStatusA');
15759: SC_LOCK', '__Pointer');
15760: _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOwner: PChar; dwLockDuration:DWORD; end';
15761: _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15762: QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15763: QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15764: TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15765: //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15766: TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15767: _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15768: +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15769: +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15770: +'iceStartName : PChar; lpDisplayName : PChar; end');
15771: _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15772: QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15773: QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15774: TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15775: TQueryServiceConfig', 'TQueryServiceConfigA');
15776: Function CloseServiceHandle( hSCObject : SC_HANDLE ) : BOOL';
15777: Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15778: Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;
15779: +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar ) : SC_HANDLE );
15780: Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;
15781: +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar ) : SC_HANDLE );
15782: Function DeleteService( hService : SC_HANDLE ) : BOOL';
15783: Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD ) : BOOL';
15784: Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState:DWORD;var lpServices:
TEnumServiceStatus;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD ):BOOL;
15785: Function GetServiceKeyName(hSCManager:SC_HANDLE;pdisplayName,lpServiceName:PChar;var
lpcchBuffer:DWORD):BOOL';
15786: Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
lpcchBuffer:DWORD):BOOL;
15787: Function LockServiceDatabase( hSCManager : SC_HANDLE ) : SC_LOCK );
15788: Function NotifyBootConfigStatus( BootAcceptable : BOOL ) : BOOL';
15789: Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD ) : SC_HANDLE );
15790: Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE );
15791: Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : BOOL';
15792: Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15793: Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15794: Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15795: Function UnlockServiceDatabase( ScLock : SC_LOCK ) : BOOL';
15796: end;
15797:
15798: procedure SIRegister_JvPickDate(CL: TPPascalCompiler);
15799: begin
15800: Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
AStrtOfWeek : TDayOfWeekName; AWeekends : TDaysOfWeek; AWeekendColor : TColor;
BtnHints : TStrings; MinDate : TDateTime; MaxDate : TDateTime ) : Boolean;
15801: Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
DlgCaption:TCaption;AStrtOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDateTime;
15802: Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime) : Boolean;
15803: Function CreatePopupCalendar(AOwner : TComponent; ABiDiMode : TBiDiMode; MinDate : TDateTime; MaxDate : TDateTime) :
TWinControl;
15804: Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStrtOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15805: Function CreateNotifyThread(const
FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15806: end;
15807:
15808: procedure SIRegister_JclNTFS2(CL: TPPascalCompiler);
15809: begin
15810: CL.AddClassN(CL.FindClass('TOBJECT'), 'EJclNtfsError');
15811: CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15812: Function NtfsGetCompression2( const FileName : TFileName; var State : Short ) : Boolean;';
15813: Function NtfsGetCompression12( const FileName : TFileName ) : TFileCompressionState;');
15814: Function NtfsSetCompression2( const FileName : TFileName; const State : Short ) : Boolean;');
15815: Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15816: Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');

```

```

15817: Procedure NtfsSetDefaultFileCompression2( const Directory : string; const State:TFileCompressionState );
15818: Procedure NtfsSetPathCompression2( const Path:string;const State:TFileCompressionState;Recursive:Bool );
15819: Function NtfsSetSparse2( const FileName : string ) : Boolean';
15820: Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64 ) : Boolean';
15821: Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64 ) : Boolean';
15822: Function NtfsSparseStreamsSupported2( const Volume : string ) : Boolean';
15823: Function NtfsGetSparse2( const FileName : string ) : Boolean';
15824: Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD ) : Boolean';
15825: Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword ) : Boolean';
15826: Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD ) : Boolean';
15827: Function NtfsReparsePointsSupported2( const Volume : string ) : Boolean';
15828: Function NtfsFileHasReparsePoint2( const Path : string ) : Boolean';
15829: Function NtfsIsFolderMountPoint2( const Path : string ) : Boolean';
15830: Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char ) : Boolean';
15831: Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString ) : Boolean';
15832: CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15833: Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15834: Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15835: Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15836: Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15837: Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped ) : Boolean';
15838: Function NtfsCreateJunctionPoint2( const Source, Destination : string ) : Boolean';
15839: Function NtfsDeleteJunctionPoint2( const Source : string ) : Boolean';
15840: Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string ) : Boolean;
15841: CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15842: +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )');
15843: CL.AddTypeS('TStreamIds', 'set of TStreamId');
15844: TInternalFindStreamData', 'record FileHandle:THandle; Context : TObject; StreamIds:TStreamIds; end');
15845: CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15846: +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15847: Function NtfsFindFirstStream2( const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData ) : Bool;
15848: Function NtfsFindNextStream2( var Data : TFindStreamData ) : Boolean';
15849: Function NtfsFindStreamClose2( var Data : TFindStreamData ) : Boolean';
15850: Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String ) : Boolean';
15851: Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString ) : Boolean';
15852: Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString ) : Boolean';
15853: CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15854: Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo ) : Boolean';
15855: Function NtfsDeleteHardLinks2( const Path:string;const FileIdxHigh,FIdxLow:Card;const List:TStrings ) : Bool;
15856: Function NtfsDeleteHardLinks2( const FileName : string ) : Boolean';
15857: FindClass('TOBJECT'), 'TJclFileSummaryError');
15858: TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15859: TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15860: TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID ) : Boolean');
15861: CL.AddClassN(CL.FindClass('TOBJECT'), 'TJclFileSummary');
15862: SIRegister_TJclFilePropertySet(CL);
15863: //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15864: SIRegister_TJclFileSummary(CL);
15865: SIRegister_TJclDocSummaryInformation(CL);
15866: SIRegister_TJclStorageSummaryInformation(CL);
15867: SIRegister_TJclMediaFileSummaryInformation(CL);
15868: SIRegister_TJclMSISummaryInformation(CL);
15869: SIRegister_TJclShellSummaryInformation(CL);
15870: SIRegister_TJclImageSummaryInformation(CL);
15871: SIRegister_TJclDisplacedSummaryInformation(CL);
15872: SIRegister_TJclBriefCaseSummaryInformation(CL);
15873: SIRegister_TJclMiscSummaryInformation(CL);
15874: SIRegister_TJclWebViewSummaryInformation(CL);
15875: SIRegister_TJclMusicSummaryInformation(CL);
15876: SIRegister_TJclDRMSummaryInformation(CL);
15877: SIRegister_TJclVideoSummaryInformation(CL);
15879: SIRegister_TJclAudioSummaryInformation(CL);
15880: SIRegister_TJclControlPanelSummaryInformation(CL);
15881: SIRegister_TJclVolumeSummaryInformation(CL);
15882: SIRegister_TJclShareSummaryInformation(CL);
15883: SIRegister_TJclLinkSummaryInformation(CL);
15884: SIRegister_TJclQuerySummaryInformation(CL);
15885: SIRegister_TJclImageInformation(CL);
15886: SIRegister_TJclJpegSummaryInformation(CL);
15887: end;
15888:
15889: procedure SIRegister_Jcl8087(CL: TPPascalCompiler);
15890: begin
15891: AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15892: T8087Rounding', '( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15893: T8087Infinity', '( icProjective, icAffine )');
15894: T8087Exception', '( emInvalidOp, emDenormalizedOperand, emZeroDivide, emOverflow, emUnderflow, emPrecision );
15895: CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15896: Function Get8087ControlWord : Word');
15897: Function Get8087Infinity : T8087Infinity');
15898: Function Get8087Precision : T8087Precision');
15899: Function Get8087Rounding : T8087Rounding');
15900: Function Get8087StatusWord( ClearExceptions : Boolean ) : Word');
15901: Function Set8087Infinity( const Infinity : T8087Infinity ) : T8087Infinity');
15902: Function Set8087Precision( const Precision : T8087Precision ) : T8087Precision');
15903: Function Set8087Rounding( const Rounding : T8087Rounding ) : T8087Rounding');
15904: Function Set8087ControlWord( const Control : Word ) : Word');
15905: Function ClearPending8087Exceptions : T8087Exceptions');

```

```

15906: Function GetPending8087Exceptions : T8087Exceptions');
15907: Function GetMasked8087Exceptions : T8087Exceptions');
15908: Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15909: Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15910: Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15911: end;
15912:
15913: procedure SIRegister_JvBoxProcs(CL: TPSPPascalCompiler);
15914: begin
15915: Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15916: Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15917: Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool';
15918: Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15919: Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15920: Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15921: Function BoxGetFirstSelection( List : TWinControl) : Integer');
15922: Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15923: end;
15924:
15925: procedure SIRegister_UrlMon(CL: TPSPPascalCompiler);
15926: begin
15927: //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15928: //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15929: CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15930: type ULONG', 'Cardinal');
15931:     LPCWSTR', 'PChar');
15932: CL.AddTypeS('LPWSTR', 'PChar');
15933: LPSTR', 'PChar');
15934: TBindVerb', 'ULONG');
15935: TBindInfoF', 'ULONG');
15936: TBindF', 'ULONG');
15937: TBSCF', 'ULONG');
15938: TBindStatus', 'ULONG');
15939: TCIPStatus', 'ULONG');
15940: TBindString', 'ULONG');
15941: TPiFlags', 'ULONG');
15942: TOIBdgFlags', 'ULONG');
15943: TParseAction', 'ULONG');
15944: TPSUAction', 'ULONG');
15945: TQueryOption', 'ULONG');
15946: TPUAF', 'ULONG');
15947: TSZMFlags', 'ULONG');
15948: TUrlZone', 'ULONG');
15949: TUrlTemplate', 'ULONG');
15950: TZAFlags', 'ULONG');
15951: TUrlZoneReg', 'ULONG');
15952: 'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15953: CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15954: const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15955: const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15956: const 'CF_NULL','LongInt').SetInt( 0);
15957: const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15958: const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15959: const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15960: const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15961: const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15962: const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15963: const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15964: const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15965: const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15966: const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15967: const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15968: const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15969: const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15970: const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15971: const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15972: const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15973: const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15974: const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15975: const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15976: const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15977: const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15978: const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15979: const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15980: const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15981: const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15982: const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15983: const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15984: const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15985: const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15986: const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15987: const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15988: const 'MK_S_ASYNCNCHRONOUS','LongWord').SetUInt( $000401E8);
15989: const 'S_ASYNCNCHRONOUS','LongWord').SetUInt( $000401E8);
15990: const 'E_PENDING','LongWord').SetUInt( $8000000A);
15991: CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15992: SIRegister_IPersistMoniker(CL);
15993: SIRegister_IBindProtocol(CL);

```

```

15994:     SIRegister_IBinding(CL);
15995:     const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15996:     const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15997:     const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15998:     const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15999:     const 'BINDINFO_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
16000:     const 'BINDINFO_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
16001:     const 'BINDF_ASYNCRONOUS','LongWord').SetUInt( $00000001);
16002:     const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
16003:     const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
16004:     const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
16005:     const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
16006:     const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
16007:     const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
16008:     const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
16009:     const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
16010:     const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
16011:     const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
16012:     const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
16013:     const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
16014:     const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
16015:     const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
16016:     const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
16017:     const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
16018:     const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
16019: //const 'BINDF_DONTUSECACHE','','').SetString( BINDF_GETNEWESTVERSION);
16020: //const 'BINDF_DONTPUTINCACHE','','').SetString( BINDF_NOWRITECACHE);
16021: //const 'BINDF_NOCOPYDATA','','').SetString( BINDF_PULLDATA);
16022: const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
16023: const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
16024: const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
16025: const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
16026: const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
16027: const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
16028: const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16029: const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16030: const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16031: const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16032: const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16033: const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
16034: const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
16035: const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
16036: const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16037: const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16038: const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16039: const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16040: const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
16041: const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
16042: const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
16043: const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16044: const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16045: const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16046: const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
16047: const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16048: const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16049: const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
16050: const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16051: const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16052: const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
16053: const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
16054: const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
16055: const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16056: const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16057: const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16058: const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16059: const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16060: const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16061: const 'BINDSTATUS_COMPACT_POLICY RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16062: const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY RECEIVED + 1);
16063: const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16064: const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
16065: const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16066: const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
16067: const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16068: const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
16069: const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16070: const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16071: const 'BINDSTATUS_SESSION_COOKIE RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16072: const 'BINDSTATUS_PERSISTENT_COOKIE RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE RECEIVED + 1);
16073: const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE RECEIVED + 1);
16074: const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16075: const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
16076: const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
16077: const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
16078: const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16079: // PBindInfo', '^TBindInfo // will not work');
16080: {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16081: +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16082: +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'

```

```

16083: +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16084: +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16085: TBindInfo', '_tagBINDINFO');
16086: BINDINFO', '_tagBINDINFO');}
16087: _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16088: +'criptor : DWORD; bInheritHandle : BOOL; end');
16089: TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
16090: REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
16091: //PREmBindInfo', '^TRemBindInfo // will not work');
16092: {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR, '
16093: +'grfBindInfo : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16094: +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16095: +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16096: +'n; dwReserved : DWORD; end');
16097: TRemBindInfo', '_tagRemBINDINFO');
16098: RemBINDINFO', '_tagRemBINDINFO');
16099: //PREmFormatEtc', '^TRemFormatEtc // will not work');
16100: tagRemFORMATETC', 'record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16101: TRemFormatEtc', 'tagRemFORMATETC');
16102: RemFORMATETC', 'tagRemFORMATETC');
16103: SIRegister_IBindStatusCallback(CL);
16104: SIRegister_IAuthenticate(CL);
16105: SIRegister_IHttpNegotiate(CL);
16106: SIRegister_IWIndowForBindingUI(CL);
16107: const 'CIP_DISK_FULL','LongInt').SetInt( 0 );
16108: const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1 );
16109: const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1 );
16110: const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1 );
16111: const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1 );
16112: const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1 );
16113: const 'CIP_EXE_SELF_REGISTRATION_TIMEOUT','LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1 );
16114: const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTRATION_TIMEOUT + 1 );
16115: const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1 );
16116: const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1 );
16117: SIRegister_ICodeInstall(CL);
16118: SIRegister_IWInetInfo(CL);
16119: const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534 );
16120: SIRegister_IHttpSecurity(CL);
16121: SIRegister_IWInetHttpInfo(CL);
16122: SIRegister_IBindHost(CL);
16123: const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001 );
16124: const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002 );
16125: const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003 );
16126: Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback : HResult );
16127: Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback : HResult );
16128: Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
: IBindStatusCallback ) : HResult );
16129: Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
IBindStatusCallback ) : HResult );
16130: Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult';
16131: Function HlinkGoBack( unk : IUnknown ) : HResult';
16132: Function HlinkGoForward( unk : IUnknown ) : HResult';
16133: Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR ) : HResult';
16134: // Function HlinkNavigateMoniker( unk : IUnknown; mkTarget : IMoniker ) : HResult );
16135: SIRegister_IInternet(CL);
16136: const 'BINDSTRING_HEADERS','LongInt').SetInt( 1 );
16137: const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1 );
16138: const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1 );
16139: const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1 );
16140: const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1 );
16141: const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1 );
16142: const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1 );
16143: const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1 );
16144: const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1 );
16145: const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1 );
16146: const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1 );
16147: const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1 );
16148: const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1 );
16149: const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1 );
16150: //POLEStrArray', '^TOLESTRArray // will not work';
16151: SIRegister_IInternetBindInfo(CL);
16152: const 'PI_PARSE_URL','LongWord').SetUInt( $00000001 );
16153: const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002 );
16154: const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004 );
16155: const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008 );
16156: const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010 );
16157: const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020 );
16158: const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040 );
16159: const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080 );
16160: const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100 );
16161: const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200 );
16162: const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000 );
16163: //const 'PI_DOCFILECLSIDLOOKUP','','').SetString( PI_CLSIDLOOKUP );
16164: //PProtocolData', '^TProtocolData // will not work');
16165: _tagPROTOCOLDATA', 'record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16166: TProtocolData', '_tagPROTOCOLDATA');
16167: PROTOCOLDATA', '_tagPROTOCOLDATA');
16168: CL.AddInterface(CL.FindInterface('IUNKNOWN'),IIInternetProtocolSink, 'IIInternetProtocolSink');

```

```

16169:     SIRegister_IInternetProtocolRoot(CL);
16170:     SIRegister_IInternetProtocol(CL);
16171:     SIRegister_IInternetProtocolSink(CL);
16172:     const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16173:     SIRegister_IInternetSession(CL);
16174:     SIRegister_IInternetThreadSwitch(CL);
16175:     SIRegister_IInternetPriority(CL);
16176:     const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16177:     const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16178:     const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16179:     const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16180:     const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16181:     const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16182:     const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16183:     const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16184:     const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16185:     const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16186:     const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16187:     const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16188:     const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16189:     const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16190:     const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16191:     const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16192:     const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16193:     const 'PSU_DEFAULT','LongInt').SetInt( 1);
16194:     const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16195:     const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16196:     const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16197:     const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16198:     const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16199:     const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16200:     const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16201:     const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16202:     const 'QUERYUSES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16203:     const 'QUERY_IS_CACHED','LongInt').SetInt( QUERYUSES_NETWORK + 1);
16204:     const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16205:     const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16206:     const 'QUERYUSES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16207:     const 'QUERY_IS_SECURE','LongInt').SetInt( QUERYUSES_CACHE + 1);
16208:     const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16209:     SIRegister_IInternetProtocolInfo(CL);
16210:     IOInet', 'IInternet');
16211:     IOInetBindInfo', 'IInternetBindInfo');
16212:     IOInetProtocolRoot', 'IInternetProtocolRoot');
16213:     IOInetProtocol', 'IInternetProtocol');
16214:     IOInetProtocolSink', 'IInternetProtocolSink');
16215:     IOInetProtocolInfo', 'IInternetProtocolInfo');
16216:     IOInetSession', 'IInternetSession');
16217:     IOInetPriority', 'IInternetPriority');
16218:     IOInetThreadSwitch', 'IInternetThreadSwitch');
16219:     Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16220:     Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16221:     Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult';
16222:     Function CoInternetGetProtocolFlags( pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult';
16223:     Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult';
16224:     Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession: IInternetSes;dwReserved:DWORD):HResult;
16225:     Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16226:     Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16227:     Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16228:     Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult';
16229:     Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult';
16230:     Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult';
16231:     Function OInetGetSession(dwSessionMode:DWORD; var pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16232:     //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult';
16233:     //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16234:     // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult( $800C0011 ) );
16235:     // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult( $800C0012 ) );
16236:     //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult( $800C0011 ) );
16237:     //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult( $800C0013 ) );
16238:     //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult( $800C0014 ) );
16239:     const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16240:     SIRegister_IInternetSecurityMgrSite(CL);
16241:     const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16242:     const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16243:     const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16244:     const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16245:     const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16246:     const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16247:     const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16248:     const 'PUAF_NOUI','LongWord').SetUInt( $00000001);

```

```

16249: const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16250: const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16251: const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16252: const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16253: const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16254: const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16255: const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16256: const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16257: const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16258: const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16259: const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16260: const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16261: const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16262: const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16263: const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16264: const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16265: const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16266: const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16267: const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16268: SIRegister_IInternetSecurityManager(CL);
16269: SIRegister_IInternetHostSecurityManager(CL);
16270: SIRegister_IInternetSecurityManagerEx(CL);
16271: const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16272: const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16273: const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16274: const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16275: const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16276: const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16277: const 'URLACTION_ACTIVE_MIN','LongWord').SetUInt( $00001200);
16278: const 'URLACTION_ACTIVE_RUN','LongWord').SetUInt( $00001200);
16279: const 'URLACTION_ACTIVE_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16280: const 'URLACTION_ACTIVE_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16281: const 'URLACTION_ACTIVE_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16282: const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16283: const 'URLACTION_ACTIVECONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16284: const 'URLACTION_ACTIVE_X_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16285: const 'URLACTION_ACTIVE_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16286: const 'URLACTION_ACTIVE_CURR_MAX','LongWord').SetUInt( $00001206);
16287: const 'URLACTION_ACTIVE_MAX','LongWord').SetUInt( $000013FF);
16288: const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16289: const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16290: const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16291: const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16292: const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16293: const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16294: const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16295: const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16296: const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16297: const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16298: const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16299: const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16300: const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16301: const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16302: const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16303: const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16304: const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16305: const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16306: const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16307: const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16308: const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16309: const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16310: const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16311: const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16312: const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16313: const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16314: const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16315: const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16316: const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16317: const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16318: const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16319: const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16320: const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16321: const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16322: const 'URLACTION_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16323: const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16324: const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16325: const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16326: const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16327: const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16328: const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16329: const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16330: const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16331: const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16332: const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16333: const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16334: const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16335: const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16336: const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16337: const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);

```

```

16338: const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS', 'LongWord').SetUInt( $00001D01);
16339: const 'URLACTION_INFODELIVERY_NO_Removing_CHANNELS', 'LongWord').SetUInt( $00001D02);
16340: const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS', 'LongWord').SetUInt( $00001D03);
16341: const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS', 'LongWord').SetUInt( $00001D04);
16342: const 'URLACTION_INFODELIVERY_NO_Removing_SUBSCRIPTIONS', 'LongWord').SetUInt( $00001D05);
16343: const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING', 'LongWord').SetUInt( $00001D06);
16344: const 'URLACTION_INFODELIVERY_CURR_MAX', 'LongWord').SetUInt( $00001D06);
16345: const 'URLACTION_INFODELIVERY_MAX', 'LongWord').SetUInt( $00001Dff);
16346: const 'URLACTION_CHANNEL_SOFTDIST_MIN', 'LongWord').SetUInt( $00001E00);
16347: const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS', 'LongWord').SetUInt( $00001E05);
16348: const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT', 'LongWord').SetUInt( $00010000);
16349: const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE', 'LongWord').SetUInt( $00020000);
16350: const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL', 'LongWord').SetUInt( $00030000);
16351: const 'URLACTION_CHANNEL_SOFTDIST_MAX', 'LongWord').SetUInt( $00001EFF);
16352: const 'URLACTION_BEHAVIOR_MIN', 'LongWord').SetUInt( $00002000);
16353: const 'URLACTION_BEHAVIOR_RUN', 'LongWord').SetUInt( $00002000);
16354: const 'URLPOLICY_BEHAVIOR_CHECK_LIST', 'LongWord').SetUInt( $00001000);
16355: const 'URLACTION_FEATURE_MIN', 'LongWord').SetUInt( $00002100);
16356: const 'URLACTION_FEATURE_MIME_SNIFFING', 'LongWord').SetUInt( $00002100);
16357: const 'URLACTION_FEATURE_ZONE_ELEVATION', 'LongWord').SetUInt( $00002101);
16358: const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS', 'LongWord').SetUInt( $00002102);
16359: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN', 'LongWord').SetUInt( $00002200);
16360: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI', 'LongWord').SetUInt( $00002200);
16361: const 'URLACTION_AUTOMATIC_ACTIVEX_UI', 'LongWord').SetUInt( $00002201);
16362: const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS', 'LongWord').SetUInt( $00002300);
16363: const 'URLPOLICY_ALLOW', 'LongWord').SetUInt( $00);
16364: const 'URLPOLICY_QUERY', 'LongWord').SetUInt( $01);
16365: const 'URLPOLICY_DISALLOW', 'LongWord').SetUInt( $03);
16366: const 'URLPOLICY_NOTIFY_ON_ALLOW', 'LongWord').SetUInt( $10);
16367: const 'URLPOLICY_NOTIFY_ON_DISALLOW', 'LongWord').SetUInt( $20);
16368: const 'URLPOLICY_LOG_ON_ALLOW', 'LongWord').SetUInt( $40);
16369: const 'URLPOLICY_LOG_ON_DISALLOW', 'LongWord').SetUInt( $80);
16370: const 'URLPOLICY_MASK_PERMISSIONS', 'LongWord').SetUInt( $0f);
16371: Function GetUrlPolicyPermissions( dw : DWORD) : DWORD';
16372: Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD';
16373: const 'URLZONE_PREDEFINED_MIN', 'LongInt').SetInt( 0);
16374: const 'URLZONE_LOCAL_MACHINE', 'LongInt').SetInt( 0);
16375: const 'URLZONE_INTRANET', 'LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16376: const 'URLZONE_TRUSTED', 'LongInt').SetInt( URLZONE_INTRANET + 1);
16377: const 'URLZONE_INTERNET', 'LongInt').SetInt( URLZONE_TRUSTED + 1);
16378: const 'URLZONE_UNTRUSTED', 'LongInt').SetInt( URLZONE_INTERNET + 1);
16379: const 'URLZONE_PREDEFINED_MAX', 'LongInt').SetInt( 999);
16380: const 'URLZONE_USER_MIN', 'LongInt').SetInt( 1000);
16381: const 'URLZONE_USER_MAX', 'LongInt').SetInt( 10000);
16382: const 'URLTEMPLATE_CUSTOM', 'LongWord').SetUInt( $00000000);
16383: const 'URLTEMPLATE_PREAMBLE', 'LongWord').SetUInt( $00010000);
16384: const 'URLTEMPLATE_LOW', 'LongWord').SetUInt( $00010000);
16385: const 'URLTEMPLATE_MEDIUM', 'LongWord').SetUInt( $00011000);
16386: const 'URLTEMPLATE_HIGH', 'LongWord').SetUInt( $00012000);
16387: const 'URLTEMPLATE_PREAMBLE_MAX', 'LongWord').SetUInt( $00020000);
16388: const 'MAX_ZONE_PATH', 'LongInt').SetInt( 260);
16389: const 'MAX_ZONE_DESCRIPTION', 'LongInt').SetInt( 200);
16390: const 'ZAFLAGS_CUSTOM_EDIT', 'LongWord').SetUInt( $00000001);
16391: const 'ZAFLAGS_ADD_SITES', 'LongWord').SetUInt( $00000002);
16392: const 'ZAFLAGS_REQUIRE_VERIFICATION', 'LongWord').SetUInt( $00000004);
16393: const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE', 'LongWord').SetUInt( $00000008);
16394: const 'ZAFLAGS_INCLUDE_INTRANET_SITES', 'LongWord').SetUInt( $00000010);
16395: const 'ZAFLAGS_NO_UI', 'LongWord').SetUInt( $00000020);
16396: const 'ZAFLAGS_SUPPORTS_VERIFICATION', 'LongWord').SetUInt( $00000040);
16397: const 'ZAFLAGS_UNC_AS_INTRANET', 'LongWord').SetUInt( $00000080);
16398: const 'ZAFLAGS_USE_LOCKED_ZONES', 'LongWord').SetUInt( $00010000);
16399: //PZoneAttributes', '_TZoneAttributes // will not work');
16400: _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');
16401: { _ZONEATTRIBUTES = packed record
16402:   cbSize: ULONG;
16403:   szDisplayName: array [0..260 - 1] of WideChar;
16404:   szDescription: array [0..200 - 1] of WideChar;
16405:   szIconPath: array [0..260 - 1] of WideChar;
16406:   dwTemplateMinLevel: DWORD;
16407:   dwTemplateRecommended: DWORD;
16408:   dwTemplateCurrentLevel: DWORD;
16409:   dwFlags: DWORD;
16410: end; }
16411: TZoneAttributes', '_ZONEATTRIBUTES');
16412: ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16413: const 'URLZONEREG_DEFAULT', 'LongInt').SetInt( 0);
16414: const 'URLZONEREG_HKLM', 'LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16415: const 'URLZONEREG_HKCU', 'LongInt').SetInt( URLZONEREG_HKLM + 1);
16416: SIRegister_IInternetZoneManager(CL);
16417: SIRegister_IInternetZoneManagerEx(CL);
16418: const 'SOFTDIST_FLAG_USAGE_EMAIL', 'LongWord').SetUInt( $00000001);
16419: const 'SOFTDIST_FLAG_USAGE_PRECACHE', 'LongWord').SetUInt( $00000002);
16420: const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL', 'LongWord').SetUInt( $00000004);
16421: const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION', 'LongWord').SetUInt( $00000008);
16422: const 'SOFTDIST_ADSTATE_NONE', 'LongWord').SetUInt( $00000000);
16423: const 'SOFTDIST_ADSTATE_AVAILABLE', 'LongWord').SetUInt( $00000001);
16424: const 'SOFTDIST_ADSTATE_DOWNLOADED', 'LongWord').SetUInt( $00000002);

```

```

16425: const 'SOFTDIST_ADSTATE_INSTALLED', 'LongWord').SetUInt( $00000003);
16426: //PCodeBaseHold', '^TCodeBaseHold // will not work');
16427: _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR;
16428: + 'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16429: TCodeBaseHold', '_tagCODEBASEHOLD');
16430: CODEBASEHOLD', '_tagCODEBASEHOLD');
16431: //PSoftDistInfo', '^TSoftDistInfo // will not work');
16432: _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd';
16433: +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI';
16434: +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS';
16435: +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve';
16436: +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16437: TSoftDistInfo', '_tagSOFTDISTINFO');
16438: SOFTDISTINFO', '_tagSOFTDISTINFO');
16439: SIRegister_ISoftDistExt(CL);
16440: Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16441: Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
dwAdvertisedVersionLS : DWORD) : HResult');
16442: SIRegister_IDataFilter(CL);
16443: // PProtocolFilterData', '^TProtocolFilterData // will not work');
16444: _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolsSink : ';
16445: +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil';
16446: +'terFlags : DWORD; end');
16447: TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16448: PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16449: //PDataInfo', '^TDataInfo // will not work');
16450: _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL';
16451: +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16452: TDataInfo', '_tagDATAINFO');
16453: DATAINFO', '_tagDATAINFO');
16454: SIRegister_IEncodingFilterFactory(CL);
16455: Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16456: //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16457: //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16458: //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16459: _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU';
16460: +'rlName : LPSTR; StartTime : TSystemTime; EndTime: TSystemTime; lpszExtendedInfo : LPSTR; end');
16461: THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16462: HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16463: Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16464: end;
16465:
16466: procedure SIRegister_DFFUtils(CL: TPPascalCompiler);
16467: begin
16468: Procedure reformatMemo( const m : TCustomMemo)');
16469: Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16470: Procedure MoveToTop( memo : TMemo)');
16471: Procedure ScrollToTop( memo : TMemo)');
16472: Function LineNumberClicked( memo : TMemo) : integer');
16473: Function MemoClickedLine( memo : TMemo) : integer');
16474: Function ClickedMemoLine( memo : TMemo) : integer');
16475: Function MemoLineClicked( memo : TMemo) : integer');
16476: Function LinePositionClicked( Memo : TMemo) : integer');
16477: Function ClickedMemoPosition( memo : TMemo) : integer');
16478: Function MemoPositionClicked( memo : TMemo) : integer');
16479: Procedure AdjustGridSize( grid : TDrawGrid)');
16480: Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16481: Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16482: Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer)');
16483: Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascendant : boolean;');
16484: Procedure sortstrDown( var s : string)');
16485: Procedure sortstrUp( var s : string)');
16486: Procedure rotatestrleft( var s : string)');
16487: Function dffstrtofloatdef( s : string; default : extended) : extended');
16488: Function deblank( s : string) : string');
16489: Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16490: Procedure FreeAndClearListBox( C : TListBox)');
16491: Procedure FreeAndClearMemo( C : TMemo)');
16492: Procedure FreeAndClearStringList( C : TStringList)');
16493: Function dffgetfilesize( f : TSearchrec) : int64');
16494: end;
16495:
16496: procedure SIRegister_MathLib(CL: TPPascalCompiler);
16497: begin
16498: CL.AddTypeS('intset', 'set of byte');
16499: TPoint64', 'record x : int64; y : int64; end');
16500: Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16501: Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16502: Function GeneratePentagon( n : integer) : integer');
16503: Function IsPentagon( p : integer) : boolean');
16504: Function isSquare( const N : int64) : boolean');
16505: Function isCube( const N : int64) : boolean');
16506: Function isPalindrome( const n : int64) : boolean');
16507: Function isPalindrome1( const n : int64; var len : integer) : boolean');
16508: Function GetEulerPhi( n : int64) : int64');
16509: Function dffIntPower( a, b : int64) : int64;');
16510: Function IntPower1( a : extended; b : int64) : extended;');
16511: Function gcd2( a, b : int64) : int64');
16512: Function GCDMany( A : array of integer) : integer');

```

```

16513: Function LCMMany( A : array of integer ) : integer');
16514: Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16515: Function dffFactorial( n : int64 ) : int64';
16516: Function digitcount( n : int64 ) : integer';
16517: Function nextpermute( var a : array of integer ) : boolean');
16518: Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16519: Function convertStringToDecimal( s : string; var n : extended ) : Boolean';
16520: Function InttoBinaryStr( nn : integer ) : string');
16521: Function StrtoAngle( const s : string; var angle : extended ) : boolean');
16522: Function AngleToStr( angle : extended ) : string');
16523: Function deg2rad( deg : extended ) : extended');
16524: Function rad2deg( rad : extended ) : extended');
16525: Function GetLongToMercProjection( const long : extended ) : extended');
16526: Function GetLatToMercProjection( const Lat : Extended ) : Extended');
16527: Function GetMercProjectionToLong( const ProjLong : extended ) : extended');
16528: Function GetMercProjectionToLat( const ProjLat : extended ) : extended');
16529: SIRegister_TPrimes(CL);
16530: //RIRegister_TPrimes(CL);
16531: //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16532: CL.AddConstantN('minmark','LongInt').SetInt( 180);
16533: Function Random64( const N : Int64 ) : Int64;');
16534: Procedure Randomize64');
16535: Function Random641 : extended;');
16536: Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUUtils
16537: end;
16538:
16539: procedure SIRegister_UGeometry(CL: TPSPPascalCompiler);
16540: begin
16541:   TrealPoint', 'record x : extended; y : extended; end');
16542:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16543:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16544:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16545:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16546:   PPResult', '( PPOutside, PPIInside, PPVertex, PPEdge, PPError )');
16547:   Function realpoint( x, y : extended ) : TRealPoint');
16548:   Function dist( const p1, p2 : TrealPoint ) : extended');
16549:   Function intdist( const p1, p2 : TPoint ) : integer');
16550:   Function dffLine( const p1, p2 : TPoint ) : Tline;');
16551:   Function Linel( const p1, p2 : TRealPoint ) : TRealline');
16552:   Function dffCircle( const cx, cy, R : integer ) : TCircle');
16553:   Function Circlel( const cx, cy, R : extended ) : TRealCircle');
16554:   Function GetTheta( const L : TLine ) : extended');
16555:   Function GetTheta1( const p1, p2 : TPoint ) : extended');
16556:   Function GetTheta2( const p1, p2 : TRealPoint ) : extended');
16557:   Procedure Extendline( var L : TLine; dist : integer );
16558:   Procedure Extendline1( var L : TRealLine; dist : extended );
16559:   Function Linesintersect( line1, line2 : TLine ) : boolean );
16560:   Function ExtendedLinesIntersect( Line1,Line2:TLine; const extendlines:bool;var IP:TPoint ):bool;
16561:   Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint ):bool;
16562:   Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint ) : boolean );
16563:   Function PointPerpendicularLine( L : TLine; P : TPoint ) : TLine );
16564:   Function PerpDistance( L : TLine; P : TPoint ) : Integer );
16565:   Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended ) : TLine );
16566:   Function AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16567:   Function PointInPoly( const p : TPoint; Points : array of TPoint ) : PPResult );
16568:   Function PolygonArea( const points:array of TPoint;const screencoordinates:boolean;var Clockwise:boolean ) : integer;
16569:   Procedure InflatePolygon( const points:array of Tpoint; var points2:array of TPoint;var area:integer;
16570:     const screenCoordinates : boolean; const inflateby : integer );
16571:   Function PolyBuiltClockwise( const points:array of TPoint;const screencoordinates:boolean ) : bool;
16572:   Function DegtoRad( d : extended ) : extended );
16573:   Function RadtoDeg( r : extended ) : extended );
16574:   Procedure TranslateLeftTo( var L : TLine; newend : TPoint );
16575:   Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint );
16576:   Procedure RotateRightEndBy( var L : TLine; alpha : extended );
16577:   Procedure RotateRightEndTo( var L : Tline; alpha : extended );
16578:   Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint ) : boolean );
16579:   Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint ) : boolean );
16580:   Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine ) : boolean );
16581:   Function CircleCircleExtTangentLines(C1,C2:TCircle;var L1,L2,PL1,PL2,TL1,TL2:TLine):Bool;
16582: end;
16583:
16584:
16585: procedure SIRegister_UAstronomy(CL: TPSPPascalCompiler);
16586: begin
16587:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGallonLat )');
16588:   TDType', '( ttLocal, ttUT, ttGST, ttLST )');
16589:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16590:   TSunrec', 'record TrueEclLon:extended;
16591:     AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
16592:     TrueAzAlt:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRPoint;SunMeanAnomaly:extended;end;
16593:     TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl : '
16594:       +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16595:       +' arth : extended; Phase : extended; end');
16596:     TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16597:       +' Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16598:     TEclipseRec', 'record Msg : string; Status : integer; FirstConta'

```

```

16597:   +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime:end');
16598:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16599:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16600:     +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16601:     +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16602:     +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16603:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
16604:     extended; UncorrectedEarthDistance : extended; GeoEclLonLat : TRpoint; CorrectedEarthDistance:extended;
16605:     ApparentRaDecl:TRPoint; end');
16606:   SIRegister_TAstronomy(CL);
16607:   Function AngleToStr( angle : extended) : string');
16608:   Function StrToAngle( s : string; var angle : extended) : boolean';
16609:   Function HoursToStr24( t : extended) : string');
16610:   Function RPoint( x, y : extended) : TRPoint');
16611:   Function getStimenename( t : TDTType) : string');
16612: end;
16613: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16614: begin
16615:   TCardValue', 'Integer');
16616:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16617:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16618:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16619:   SIRegister_TCard(CL);
16620:   SIRegister_TDeck(CL);
16621: end;
16622: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16623: begin
16624:   tMethodCall', 'Procedure');
16625:   tVerboseCall', 'Procedure ( s : string)');
16626:   // PTEdge', '^TEdge // will not work');
16627:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer;
16628:     +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16629:   SIRegister_TNode(CL);
16630:   SIRegister_TGraphList(CL);
16631: end;
16632: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16633: begin
16634:   ParserFloat', 'extended');
16635:   //PParserFloat', '^ParserFloat // will not work');
16636:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16637:   +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16638:   //POperation', '^TOperation // will not work');
16639:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16640:     +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure';
16641:     +'; Token : TDFFToken; end');
16642:   TMathProcedure', 'procedure(AnOperation: TDFFOperation)');
16643:   (CL.FindClass('TOBJECT'),'EMathParserError');
16644:   CL.FindClass('TOBJECT','ESyntaxError');
16645:   CL.FindClass('TOBJECT','EExpressionHasBlanks');
16646:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16647:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16648:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16649:   (CL.FindClass('TOBJECT'),'EBadName');
16650:   (CL.FindClass('TOBJECT'),'EParseInternalError');
16651:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16652:   SIRegister_TCustomParser(CL);
16653:   SIRegister_TExParser(CL);
16654: end;
16655: function isService: boolean;
16656: begin
16657:   result:= NOT(Application is TApplication);
16658:   {result:= Application is TServiceApplication;}
16659: end;
16660: function isApplication: boolean;
16661: begin
16662:   result:= Application is TApplication;
16663: end;
16664: //SM_REMOTESESSION = $1000
16665: function isTerminalSession: boolean;
16666: begin
16667:   result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16668: end;
16669: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16670: begin
16671:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16672:     +'String; margin_bottom : String; margin_left : String; margin_right : String'
16673:     +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16674:   Function cyURLEncode( const S : string ) : string');
16675:   Function MakeResourceURL( const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16676:   Function MakeResourceURL1( const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;
16677:   Function cyColorToHtml( aColor : TColor ) : String';
16678:   Function HtmlToColor( aHtmlColor : String ) : TColor';
16679:   //Function GetStreamEncoding( aStream : TStream ) : TEncoding';
16680:   // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding ) : Boolean');
16681: 
```

```

16684: Function AddHtmlUnicodePrefix( aHtml : String ) : String';
16685: Function RemoveHtmlUnicodePrefix( aHtml : String ) : String';
16686: Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup );
16687: Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup );
16688: CL.AddConstantN('IEBodyBorderless','String').SetString('none');
16689: CL.AddConstantN('IEBodySingleBorder','String').SetString('');
16690: CL.AddConstantN('IEDesignModeOn','String').SetString('On');
16691: CL.AddConstantN('IEDesignModeOff','String').SetString('Off');
16692: CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF );
16693: CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE );
16694: end;
16695:
16696:
16697: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16698: begin
16699:   CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600 );
16700:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16701:     +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16702:     +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16703:     +'inationsRepeat, CombinationsRepeatDown )');
16704:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16705:   SIRegister_TComboSet(CL);
16706: end;
16707:
16708: procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16709: begin
16710:   TcyCommandType', '( ctDeveloperDefined, ctUserDefined )');
16711:   TStreamContentType', '( scDeveloperDefined, scUserDefined, scString )');
16712:   TProcOnReceiveCommand', 'Procedure ( Sender: TObject; aCommand: Word; userParam : Integer )';
16713:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16714:     +'mmHandle : THandle; aString : String; userParam : Integer )';
16715:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16716:     +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer )';
16717:   SIRegister_TcyBaseComm(CL);
16718:   CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1 );
16719:   CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99 );
16720:   Function ValidatefileMappingName( aName : String ) : String';
16721:   procedure makeCaption(leftSide, Rightside:string; form:TForm);
16722:   end;
16723:
16724: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16725: begin
16726:   CL.AddTypeS('DERString', 'String');
16727:   CL.AddTypeS('DERChar', 'Char');
16728:   CL.AddTypeS('TEElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16729:     +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16730:   CL.AddTypeS('TEElementsTypes', 'set of TEElementsType');
16731:   CL.AddTypeS('DERNString', 'String');
16732:   const DERDecimalSeparator','String').SetString( '.' );
16733:   const DERDefaultChars','String')(+@/%
16734:   _.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16735:   const DERDefaultChars','String').SetString( '/%-.0123456789abcdefghijklmnopqrstuvwxyz' );
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 DERToDERCharset( aDERStr : DERString ) : DERNString';
16750:   Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean ) : String';
16751:   Function DERExtractWebMail( aDERStr : DERString ) : String';
16752:   Function DERExtractPhoneNr( aDERStr : DERString ) : String';
16753:   Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
16754:   SmartWebsiteRecognition:Boolean):TElementsType;
16755:   Function DERExecutel(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
16756:   RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
16757:   SmartWebsiteRecognition:Boolean):TElementsType;
16758:   Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
16759:   aElementsType : TEElementsType ) : String;');
16760:   Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
16761:   Boolean; var RsltDERStr : DERString; var RsltElementType : TEElementsType );');
16762:   end;
16763:   procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16764:   begin
16765:     pRGBQuadArray', '^TRGBQuadArray // will not work';
16766:     Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer';
16767:     Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean );
16768:     Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean );
16769:     Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean );
16770:     Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean );

```

```

16767: Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)'');
16768: Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16769: Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean)');'
16770: Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool');'
16771: Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
SingleDestinationColor, RefreshBmp : Boolean)');'
16772: Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');'
16773: Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');'
16774: Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16775: Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16776: Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean)');'
16777: Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16778: Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');'
16779: Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');'
16780: end';
16781:
16782: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16783: begin
16784:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16785:   TPCMChannel', '( cMono, cStereo )');
16786:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16787:   TPCMBytesPerSample', '( bs8Bit, bs16Bit )');
16788:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono16b'
16789:     +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16790:     +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16791:     +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16792:     +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16793:     +'it48000Hz, Stereo16bit48000Hz )');
16794:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16795:     +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word; ecbSiz: Word; end');
16796:   tWaveFormatEx', 'PWaveFormatEx');
16797:   HMMIO', 'Integer');
16798:   TWaveDeviceFormats', 'set of TPCMFormat');
16799:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16800:     +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16801:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16802:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16803:   TWaveOutOptions', 'set of TWaveOutOption');
16804:   TStreamOwnership2', '( soReference, soOwned )');
16805:   TWaveStreamState', '( wsReady, wssReading, wssWriting, wssWritingEx )');
16806: // PRawWave', '^TRawWave // will not work');
16807:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16808:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioError');
16809:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioSysError');
16810:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioInvalidOperation');
16811:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16812:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16813:     +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16814:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16815:     +'fee : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16816:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16817:     +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16818:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16819:     +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16820:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16821:   TWaveAudioFilterEvent', 'Procedure ( Sender : TObject; const Buffer:TObject; BufferSize:DWORD)');
16822:   GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16823:   CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16824:   CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16825:   GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16826:   CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16827:   OpenStreamWaveAudio( Stream : TStream ) : HMMIO');
16828:   CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD ) : DWORD');
16829:   GetAudioFormat( FormatTag : Word ) : String');
16830:   GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx ) : String');
16831:   GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD ) : DWORD');
16832:   GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx ) : DWORD');
16833:   GetWaveAudioPeakLevel( const Data : TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx) : Integer');
16834:   InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx) : Boolean');
16835:   SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx) : Boolean');
16836:   ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16837:   MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
TObject; BufferSize : DWORD ) : Boolean');
16838:   ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD ) : Boolean');
16839:   SetPCMFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
TPCMSamplesPerSec; BitsPerSample : TPCMBytesPerSample)');
16840:   Procedure SetPCMFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat ) );
16841:   GetPCMFormat( const pWaveFormat : PWaveFormatEx ) : TPCMFormat );
16842:   GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD ) : DWORD );
16843:   MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat ) : String );
16844:   WaitForSyncObject( SyncObject : THandle; Timeout : DWORD ) : DWORD );
16845: //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD ) : LRESULT );
16846: end;
16847:
16848: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);

```

```

16849: begin
16850:   'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16851:   CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16852:   'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
16853:   'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFFF ) );
16854:   'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
16855:   'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
16856:   'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
16857:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16858:   CL.AddTypeS('TPipeType', '( ptyp_BytByte, ptyp_MsgByte, ptyp_MsgMsg )');
16859:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16860:   SIRegister_TNamedPipe(CL);
16861:   SIRegister_TServePipe(CL);
16862:   SIRegister_TCClientPipe(CL);
16863:   Function CalculateTimeout( aBasis : DWORD ) : DWORD';
16864:   Function HasOverlappedIoCompleted( const ov : OVERLAPPED ) : Boolean';
16865:   Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean): TOverlappedResult;
16866:   Function GetStreamAsText( stm : TStream ) : string';
16867:   Procedure SetStreamAsText( const aTxt : string; stm : TStream )';
16868: end;
16869:
16870: procedure SIRegister_DPUtils(CL: TPSPPascalCompiler);
16871: begin
16872:   // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16873:   SIRegister_TThumbData(CL);
16874:   'PIC_BMD','LongInt').SetInt( 0);
16875:   'PIC_JPG','LongInt').SetInt( 1);
16876:   'THUMB_WIDTH','LongInt').SetInt( 60);
16877:   'THUMB_HEIGHT','LongInt').SetInt( 60);
16878:   Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean';
16879:   Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)';
16880:   Function ReadBitmap( Filehandle : Integer; Width, Height : Integer ) : TBitmap';
16881:   Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap )';
16882:   Function OpenPicture( fn : string; var tp : Integer ) : Integer';
16883:   Function ConvertPicture( pi : Integer; tp : Integer ) : TBitmap';
16884:   Function LoadPicture( fn : string; var w, h : Integer ) : TBitmap';
16885:   Function TurnBitmap( bmp : TBitmap; ang : Integer ) : TBitmap';
16886:   Function RotateBitmap( Bitmap : TBitmap; Direction : Integer ) : TBitmap';
16887:   Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap ) : TRect';
16888:   Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer ) : TBitmap';
16889:   Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer )';
16890:   Procedure FindFiles( path, mask : string; items : TStringList )';
16891:   Function LetFileName( s : string ) : string';
16892:   Function LetParentPath( path : string ) : string';
16893:   Function AddBackSlash( path : string ) : string';
16894:   Function CutBackSlash( path : string ) : string';
16895: end;
16896:
16897: procedure SIRegister_CommonTools(CL: TPSPPascalCompiler);
16898: begin
16899:   //BYTES','LongInt').SetInt( 1);
16900:   'KBYTES','LongInt').SetInt( 1024);
16901:   'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ) );
16902:   'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ) );
16903:   'DBG_GONE','LongWord').SetUInt( $99AC1D99);
16904:   'SHELL_NS_MYCOMPUTER','String').SetString( ':{20D04FE0-3AEA-1069-A2D8-08002B30309D}' );
16905:   SIRegister_MakeComServerMethodsPublic(CL);
16906:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16907:   Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD ) : Boolean';
16908:   Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean )';
16909:   Function TBGetTempFolder : string';
16910:   Function TBGetTempFile : string';
16911:   Function TBGetModuleFilename : string';
16912:   Function FormatModuleVersionInfo( const aFilename : string ) : string';
16913:   Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD ) : string';
16914:   Function TBGetFileSize( aFile : string; aMultipleOf : Integer ) : Integer';
16915:   Function FormatAttribString( aAttr : Integer ) : string';
16916:   Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo ) : string';
16917:   Function ShellRecycle( aWnd : HWND; aFileOrFolder : string ) : Boolean';
16918:   Function IsDebuggerPresent : BOOL';
16919:   Function TBNotImplemented : HRESULT';
16920: end;
16921:
16922: procedure SIRegister_D2_VistaHelperU(CL: TPSPPascalCompiler);
16923: begin
16924:   //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16925:   //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16926:   CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16927:   CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean;');
16928:   //TDrivesProperty = array['A'..'Z'] of boolean;
16929:   Function TBSetSystemTime( DateTime : TDateTime; DOW : word ) : boolean';
16930:   Function IsElevated : Boolean';
16931:   Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:GUID;const aIID:GUID;out aObj:TObject);
16932:   CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined )');
16933:   Function TrimNetResource( UNC : string ) : string';
16934:   Procedure GetFreeDrives( var FreeDrives : TDrivesProperty )';
16935:   Procedure GetMappedDrives( var MappedDrives : TDrivesProperty )';
16936:   Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
UserUsage : TPasswordUsage; User : string; Comment : string ) : boolean';

```

```

16937: Function UnmapDrive( Drive : char; Force : boolean) : boolean';
16938: Function TBIsWindowsVista : Boolean';
16939: Procedure SetVistaFonts( const AForm : TForm)');
16940: Procedure SetVistaContentFonts( const AFont : TFont)');
16941: Function GetProductType( var sType : String) : Boolean';
16942: Function lstrcmp( lpString1, lpString2 : PChar) : Integer';
16943: Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16944: Function lstrcpy( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar';
16945: Function lstrcpy( lpString1, lpString2 : PChar) : PChar';
16946: Function lstrcat( lpString1, lpString2 : PChar) : PChar';
16947: Function lstrlen( lpString : PChar) : Integer');
16948: Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
TokenInformation : __Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL';
16949: Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
TokenInformation : __Pointer; TokenInformationLength : DWORD) : BOOL';
16950: end;
16951:
16952: procedure SIRегистre_dwsXPlatform(CL: TPPSPascalCompiler);
16953: begin
16954:   'cLineTerminator','Char').SetString( #10);
16955:   'clineTerminators','String').SetString( #13#10);
16956:   'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD( - 1 ) );
16957:   SIRегистre_TFixedCriticalSection(CL);
16958:   SIRегистre_TMultiReadSingleWrite(CL);
16959:   Procedure SetDecimalSeparator( c : Char)';
16960:   Function GetDecimalSeparator : Char';
16961:   TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16962:   Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16963:   CL.AddTypeS('NativeInt', 'Integer');
16964:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16965:   CL.AddTypeS('NativeUIInt', 'Cardinal');
16966:   //CL.AddTypeS('PNativeUIInt', '^NativeUIInt // will not work');
16967:   //CL.AddTypeS('TBytes', 'array of Byte');
16968:   CL.AddTypeS('RawByteString', 'UnicodeString');
16969:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16970:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16971:   SIRегистre_TPath(CL);
16972:   SIRегистre_TFile(CL);
16973:   SIRегистre_TDwsThread(CL);
16974:   Function GetSystemMilliseconds : Int64';
16975:   Function UTCDateTime : TDateTime';
16976:   Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString';
16977:   Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer';
16978:   Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer';
16979:   Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer';
16980:   Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer';
16981:   Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer';
16982:   Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString';
16983:   Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString';
16984:   Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer';
16985:   Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean';
16986:   Function InterlockedIncrement( var val : Integer) : Integer';
16987:   Function InterlockedDecrement( var val : Integer) : Integer';
16988:   Procedure FastInterlockedIncrement( var val : Integer)');
16989:   Procedure FastInterlockedDecrement( var val : Integer)');
16990:   Function InterlockedExchangePointer( var target : __Pointer; val : __Pointer) : __Pointer';
16991:   Procedure SetThreadName( const threadName : Char; threadID : Cardinal)';
16992:   Procedure dwsOutputDebugString( const msg : UnicodeString)';
16993:   Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16994:   Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16995:   Procedure VarCopy( out dest : Variant; const src : Variant)');
16996:   Function VarToUnicodeStr( const v : Variant) : UnicodeString';
16997:   Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString';
16998:   Function LoadTextFromStream( aStream : TStream) : UnicodeString';
16999:   Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString';
17000:   Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)';
17001:   Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle';
17002:   Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle';
17003:   Procedure CloseFileHandle( hFile : THandle)';
17004:   Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean';
17005:   Function FileMove( const existing, new : UnicodeString) : Boolean';
17006:   Function dwsFileDelete( const fileName : String) : Boolean';
17007:   Function FileRename( const oldName, newName : String) : Boolean';
17008:   Function dwsFileSize( const name : String) : Int64';
17009:   Function dwsFileDateTime( const name : String) : TDateTime';
17010:   Function DirectSet8087CW( newValue : Word) : Word';
17011:   Function DirectSetMXCSR( newValue : Word) : Word';
17012:   Function TtoObject( const T: byte) : TObject';
17013:   Function TtoPointer( const T: byte) : __Pointer';
17014:   Procedure GetMemForT(var T: byte; Size : integer)';
17015:   Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer';
17016: end;
17017:
17018: procedure SIRегистre_AdSocket(CL: TPPSPascalCompiler);
17019: begin
17020:   'IPStrSize','LongInt').SetInt( 15 );
17021:   'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711 );
17022:   'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712 );

```

```

17023: 'ADWSBASE', 'LongInt').SetInt( 9000);
17024: CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
17025:   +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
17026: SIRegister_TApdSocketException(CL);
17027: TWsMode', '( wsClient, wsServer )');
17028: TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket)');
17029: TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrorCode : Integer)');
17030: SIRegister_TApdSocket(CL);
17031: end;
17032:
17033: procedure SIRegister_AdPort(CL: TPSPPascalCompiler);
17034: begin
17035:   SIRegister_TApdCustomComPort(CL);
17036:   SIRegister_TApdComPort(CL);
17037:   Function ComName( const ComNumber : Word ) : shortString';
17038:   Function SearchComPort( const C : TComponent ) : TApdCustomComPort');
17039: end;
17040:
17041: procedure SIRegister_PathFunc(CL: TPSPPascalCompiler);
17042: begin
17043:   Function inAddBackslash( const S : String ) : String';
17044:   Function PathChangeExt( const Filename, Extension : String ) : String';
17045:   Function PathCharCompare( const S1, S2 : PChar ) : Boolean';
17046:   Function PathCharIsSlash( const C : Char ) : Boolean';
17047:   Function PathCharIsTrailByte( const S : String; const Index : Integer ) : Boolean';
17048:   Function PathCharLength( const S : String; const Index : Integer ) : Integer';
17049:   Function inPathCombine( const Dir, Filename : String ) : String';
17050:   Function PathCompare( const S1, S2 : String ) : Integer';
17051:   Function PathDrivePartLength( const Filename : String ) : Integer';
17052:   Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;
17053:   Function inPathExpand( const Filename : String ) : String';
17054:   Function PathExtensionPos( const Filename : String ) : Integer';
17055:   Function PathExtractDir( const Filename : String ) : String';
17056:   Function PathExtractDrive( const Filename : String ) : String';
17057:   Function PathExtractExt( const Filename : String ) : String';
17058:   Function PathExtractName( const Filename : String ) : String';
17059:   Function PathExtractPath( const Filename : String ) : String';
17060:   Function PathIsRooted( const Filename : String ) : Boolean';
17061:   Function PathLastChar( const S : String ) : PChar';
17062:   Function PathLastDelimiter( const Delimiters, S : string ) : Integer';
17063:   Function PathLowercase( const S : String ) : String';
17064:   Function PathNormalizeSlashes( const S : String ) : String';
17065:   Function PathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
17066:   Function PathPos( Ch : Char; const S : String ) : Integer';
17067:   Function PathStartsWith( const S, AStartsWith : String ) : Boolean';
17068:   Function PathStrNextChar( const S : PChar ) : PChar';
17069:   Function PathStrPrevChar( const Start, Current : PChar ) : PChar';
17070:   Function PathStrScan( const S : PChar; const C : Char ) : PChar';
17071:   Function inRemoveBackslash( const S : String ) : String';
17072:   Function RemoveBackslashUnlessRoot( const S : String ) : String';
17073: Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean)');
17074: end;
17075:
17076:
17077: procedure SIRegister_CmnFunc2(CL: TPSPPascalCompiler);
17078: begin
17079:   NEWREGSTR_PATH_SETUP', 'String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17080:   NEWREGSTR_PATH_EXPLORER', 'String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17081:   NEWREGSTR_PATH_SPECIAL_FOLDERS', 'String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17082:   NEWREGSTR_PATH_UNINSTALL', 'String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17083:   NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME', 'String').SetString( 'DisplayName');
17084:   NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE', 'String').SetString( 'UninstallString');
17085: KEY_WOW64_64KEY', 'LongWord').SetUInt( $0100);
17086: //CL.AddTypeS('TLeadByteSet', 'TLeadByteSet // will not work');
17087: CL.AddTypeS('TLeadByteSet', 'set of Char');
17088: SIRegister_TOneShotTimer(CL);
17089: CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17090: 'RegViews64Bit', 'LongInt').Value.ts32 := ord(rv64bit);
17091: Function NewFileExists( const Name : String ) : Boolean';
17092: Function inDirExists( const Name : String ) : Boolean';
17093: Function FileOrDirExists( const Name : String ) : Boolean';
17094: Function IsDirectoryAndNotReparsePoint( const Name : String ) : Boolean';
17095: Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String';
17096: Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17097: Function GetIniBool( const Section,Key:String; const Default:Boolean;const Filenam:String): Boolean';
17098: Function IniKeyExists( const Section, Key, Filename : String ) : Boolean';
17099: Function IsIniSectionEmpty( const Section, Filename : String ) : Boolean';
17100: Function SetIniString( const Section, Key, Value, Filename : String ) : Boolean';
17101: Function SetIni( const Section,Key:String;const Value: Longint;const Filenam: String):Boolean';
17102: Function SetIniBool( const Section,Key:String;const Value: Boolean;const Filenam: String):Boolean';
17103: Procedure DeleteIniEntry( const Section, Key, Filename : String );
17104: Procedure DeleteIniSection( const Section, Filename : String );
17105: Function GetEnv( const EnvVar : String ) : String';
17106: Function GetCmdTail : String';
17107: Function GetCmdTailEx( StartIndex : Integer ) : String';
17108: Function NewParamCount : Integer';
17109: Function NewParamStr( Index : Integer ) : string';
17110: Function AddQuotes( const S : String ) : String';
17111: Function RemoveQuotes( const S : String ) : String';

```

```

17112: Function inGetShortName( const LongName : String ) : String';
17113: Function inGetWinDir : String');
17114: Function inGetSystemDir : String');
17115: Function GetSysWow64Dir : String');
17116: Function GetSysNativeDir( const IsWin64 : Boolean ) : String');
17117: Function inGetTempDir : String');
17118: Function StringChange( var S : String; const FromStr,ToStr : String ) : Integer');
17119: Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17120: Function AdjustLength( var S : String; const Res : Cardinal ) : Boolean');
17121: Function UsingWinNT : Boolean');
17122: Function ConvertConstPercentStr( var S : String ) : Boolean');
17123: Function ConvertPercentStr( var S : String ) : Boolean');
17124: Function ConstPos( const Ch : Char; const S : String ) : Integer');
17125: Function SkipPastConst( const S : String; const Start : Integer ) : Integer');
17126: Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String ) : Boolean');
17127: Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String ):Boolean;
17128: Function RegValueExists( H : HKEY; Name : PChar ) : Boolean');
17129: 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;
17130: Function RegOpenKeyExView(const RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17131: Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar ) : Longint;
17132: Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17133: Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyName:PChar):Longint;
17134: Function GetShellFolderPath( const FolderID : Integer ) : String');
17135: Function IsAdminLoggedOn : Boolean');
17136: Function IsPowerUserLoggedOn : Boolean');
17137: Function IsMultiByteString( const S : AnsiString ) : Boolean');
17138: Function FontExists( const FaceName : String ) : Boolean');
17139: //Procedure FreeAndNil( var Obj );
17140: Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT ) : HMODULE');
17141: Function GetUILanguage : LANGID');
17142: Function RemoveAccelChar( const S : String ) : String');
17143: Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17144: Function AddPeriod( const S : String ) : String');
17145: Function GetExceptMessage : String');
17146: Function GetPreferredUIFont : String');
17147: Function IsWildcard( const Pattern : String ) : Boolean');
17148: Function WildcardMatch( const Text, Pattern : PChar ) : Boolean');
17149: Function IntMax( const A, B : Integer ) : Integer');
17150: Function Win32ErrorString( ErrorCode : Integer ) : String');
17151: Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet );
17152: Function inCompareMem( P1, P2 : TObject; Length : Integer ) : Boolean');
17153: Function DeleteDirTree( const Dir : String ) : Boolean');
17154: Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
17155: Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT );
17156: // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17157: Function inCharInSet( C : Char; const CharSet : TSysCharSet ) : Boolean');
17158: Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String ) : Boolean');
17159: Function ShutdownBlockReasonDestroy( Wnd : HWND ) : Boolean');
17160: Function TryStrToBoolean( const S : String; var BoolResult : Boolean ) : Boolean');
17161: Procedure WaitMessageWithTimeout( const Milliseconds : DWORD );
17162: Function MoveFileReplace(const ExistingFileName, NewFileName : String ) : Boolean');
17163: Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND );
17164: end;
17165:
17166: procedure SIRегистre_CmnFunc(CL: TPPascalCompiler);
17167: begin
17168:   SIRегистre_TWindowDisabler(CL);
17169:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17170:   TMsgBoxCallbackFunc', 'procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17171:   Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox );
17172:   Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17173:   Function AppMessageBox( const Text, Caption : PChar; Flags : Longint ) : Integer');
17174:   Function MsgBoxP( const Text,Caption: PChar; const Typ: TMMsgBoxType;const Buttons:Cardinal):Int;
17175:   Function inMsgBox(const Text,Caption:String; const Typ: TMMsgBoxType;const Buttons:Cardinal):Int;
17176:   Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
Typ:TMMsgBoxType;const Buttons:Cardinal):Integer );
17177:   Procedure ReactivateTopWindow );
17178:   Procedure SetMessageBoxCaption( const Typ : TMMsgBoxType; const NewCaption : PChar );
17179:   Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean );
17180:   Procedure SetMessageBoxCallbackFunc(const AFunc : TMMsgBoxCallbackFunc; const AParam : LongInt );
17181: end;
17182:
17183: procedure SIRегистre_ImageGrabber(CL: TPPascalCompiler);
17184: begin
17185:   SIRегистre_TImageGrabber(CL);
17186:   SIRегистre_TCaptureDrivers(CL);
17187:   SIRегистre_TCaptureDriver(CL);
17188: end;
17189:
17190: procedure SIRегистre_SecurityFunc(CL: TPPascalCompiler);
17191: begin
17192:   Function GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:String;const
Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean );
17193:   Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean );
17194: end;

```

```

17195:
17196: procedure SIRегистер_RedirFunc(CL: TPSPascalCompiler);
17197: begin
17198:   CL.AddTypeS('TPreviousFsRedirectionState', 'record DidDisable : Boolean; OldValue : __Pointer; end');
17199:   Function AreFsRedirectionFunctionsAvailable : Boolean';
17200:   Function DisableFsRedirectionIf(const Disable : Boolean; var PreviousState:TPreviousFsRedirectionState):Bool;
17201:   Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState );
17202:   Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String ) : BOOL';
17203:   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';
17204:   Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
FailIfExists : BOOL) : BOOL';
17205:   Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String ) : BOOL';
17206:   Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String ) : Boolean';
17207:   Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String ) : Boolean';
17208:   Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
TWin32FindData ) : THandle';
17209:   Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String ) : DWORD';
17210:   Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String ) : String';
17211:   Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
TFileVersionNumbers ) : Boolean';
17212:   Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17213:   Function MoveFileRedir( const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
17214:   Function MoveFileExRedir( const DisableFsRedir:Bool;const ExistingFilen,NewFilename:String;const
Flags:DWORD ):BOOL;
17215:   Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String ) : Boolean';
17216:   Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String ) : BOOL';
17217:   Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17218:   Function SetNTFSCompressionRedir( const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17219:     SIRегистер_TFileRedir(CL);
17220:     SIRегистер_TTextFileReaderRedir(CL);
17221:     SIRегистер_TTextFileWriterRedir(CL);
17222:   end;
17223:
17224: procedure SIRегистер_Int64Em(CL: TPSPascalCompiler);
17225: begin
17226:   //CL.AddTypeS('LongWord', 'Cardinal');
17227:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17228:   Function Compare64( const N1, N2 : Integer64 ) : Integer';
17229:   Procedure Dec64( var X : Integer64; N : LongWord );
17230:   Procedure Dec6464( var X : Integer64; const N : Integer64 );
17231:   Function Div64( var X : Integer64; const Divisor : LongWord ) : LongWord';
17232:   Function Inc64( var X : Integer64; N : LongWord ) : Boolean';
17233:   Function Inc6464( var X : Integer64; const N : Integer64 ) : Boolean';
17234:   Function Integer64ToStr( X : Integer64 ) : String';
17235:   Function Mod64( const X : Integer64; const Divisor : LongWord ) : LongWord';
17236:   Function Mul64( var X : Integer64; N : LongWord ) : Boolean';
17237:   Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64 );
17238:   Procedure Shr64( var X : Integer64; Count : LongWord );
17239:   Function StrToInteger64( const S : String; var X : Integer64 ) : Boolean';
17240: end;
17241:
17242: procedure SIRегистер_InstFunc(CL: TPSPascalCompiler);
17243: begin
17244:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17245:   SIRегистер_TSsimpleStringList(CL);
17246:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle )');
17247:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17248:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17249:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17250:   // TMD5Digest = array[0..15] of Byte;
17251:   // TSHA1Digest = array[0..19] of Byte;
17252:   Function CheckForMutexes( Mutexes : String ) : Boolean';
17253:   Function CreateTempDir : String';
17254:   Function DecrementSharedCount( const RegView : TRegView; const Filename : String ) : Boolean';
17255:   Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
FirstRetryDelayMS, SubsequentRetryDelayMS : Integer );
17256:   //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
TDeleteFileProc; const Param : Pointer ) : Boolean';
17257:   //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer ) :
TDetermineDefaultLanguageResult';
17258:   //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17259:   Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String ) : Boolean';
17260:   Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17261:   Function GetComputerNameString : String';
17262:   Function GetFileDateTime( const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime ):Bool;
17263:   Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String ) : TMD5Digest';
17264:   Function GetMD5OfAnsiString( const S : AnsiString ) : TMD5Digest';
17265:   // Function GetMD5OfUnicodeString( const S : UnicodeString ) : TMD5Digest';
17266:   Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest';
17267:   Function GetSHA1OfAnsiString( const S : AnsiString ) : TSHA1Digest';
17268:   // Function GetSHA1OfUnicodeString( const S : UnicodeString ) : TSHA1Digest';
17269:   Function GetRegRootKeyName( const RootKey : HKEY ) : String';
17270:   Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
TotBytes:Int64):Bool;

```

```

17271: Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
17272:   TotalBytes: Integer64):Bool;
17273: Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17274: //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
17275:   String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
17276:   ResultCode:Integer) : Boolean';
17275: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
17276:   TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17277: Procedure InternalError( const Id : String)');
17278: Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17279: Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean';
17280: Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17281: Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17282: Procedure RaiseFunctionFailedError( const FunctionName : String)');
17283: Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17284: Procedure RefreshEnvironment ')';
17285: Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17286: Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17287: Procedure UnregisterFont( const FontName, FontFilename : String)');
17288: Function RestartComputer : Boolean)';
17289: Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17290: Procedure SplitNewParamStr( const Index: Integer; var AName, AValue : String)');
17291: Procedure Win32ErrorMsg( const FunctionName : String)');
17292: Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17293: Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17294: //from Func2
17295: //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
17296:   Iconfilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean;
17297:   //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String';
17298: Procedure RegisterTypeLibrary( const Filename : String)');
17299: //Procedure UnregisterTypeLibrary( const Filename : String)');
17300: function getVersionInfoEx3: TOSVersionInfoEx)';
17301: Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17302: procedure InitOle();
17303: Function ExpandConst( const S : String) : String');
17304: Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17305: Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17306: Function ExpandConstIfPrefixed( const S : String) : String');
17307: Procedure LogWindowsVersion)';
17308: Function EvalCheck( const Expression : String) : Boolean)';
17309: end;
17310:
17311: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17312: begin
17313:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TResourceDetails');
17314:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17315:   SIRegister_TResourceModule(CL);
17316:   SIRegister_TResourceDetails(CL);
17317:   SIRegister_TAnsiResourceDetails(CL);
17318:   SIRegister_TUnicodeResourceDetails(CL);
17319:   Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17320:   Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17321:   Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17322:   Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17323:   Function ResourceNameToInt( const s : string) : Integer');
17324:   Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer)';
17325: end;
17326:
17327:
17328: procedure SIRegister_TSsimpleComPort(CL: TPSPascalCompiler);
17329: begin
17330:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17331:   with CL.AddClassN(CL.FindClass('TObject'), 'TSimpleComPort') do begin
17332:     RegisterMethod('Constructor Create');
17333:     RegisterMethod('Procedure Free');
17334:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17335:     RegisterMethod('Procedure WriteString( const S : String)');
17336:     RegisterMethod('Procedure ReadString( var S : String)');
17337:   end;
17338:   Ex:= SimpleComPort:= TSsimpleComPort.Create;
17339:   SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17340:   SimpleComPort.WriteString(AsciiChar);
17341: end;
17342:
17343:
17344: procedure SIRegister_Console(CL: TPSPascalCompiler);
17345: begin
17346:   CL.AddConstantN('White','LongInt').SetInt( 15); }
17347: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17348: ('conBW40','LongInt').SetInt( 0);
17349: ('conCO40','LongInt').SetInt( 1);
17350: ('conBW80','LongInt').SetInt( 2);
17351: ('conCO80','LongInt').SetInt( 3);
17352: ('conMono','LongInt').SetInt( 7);
17353: CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17354: //CL.AddConstantN('C40','').SetString( CO40);

```

```

17355: //CL.AddConstantN('C80','').SetString( C080);
17356: Function con.ReadKey : Char');
17357: Function conKeyPressed : Boolean');
17358: Procedure conGotoXY( X, Y : Smallint)');
17359: Function conWhereX : Integer');
17360: Function conWhereY : Integer');
17361: Procedure conTextColor( Color : Byte);');
17362: Function conTextColor1 : Byte;');
17363: Procedure conTextBackground( Color : Byte);');
17364: Function conTextBackground1 : Byte);');
17365: Procedure conTextMode( Mode : Word)');
17366: Procedure conLowVideo');
17367: Procedure conHighVideo');
17368: Procedure conNormVideo');
17369: Procedure conClrScr');
17370: Procedure conClrEol');
17371: Procedure conInsLine');
17372: Procedure conDelLine');
17373: Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17374: Function conScreenWidth : Smallint');
17375: Function conScreenHeight : Smallint');
17376: Function conBufferWidth : Smallint');
17377: Function conBufferHeight : Smallint');
17378: procedure InitScreenMode;');
17379: end;
17380:
17381: (*-----*)
17382: procedure SIRegister_testutils(CL: TPSPascalCompiler);
17383: begin
17384:   SIRegister_TNoRefCountObject(CL);
17385:   Procedure FreeObjects( List : TFPList)');
17386:   Procedure GetMethodList( AObject : TObject; AList : TStrings)');
17387:   Procedure GetMethodList1( AClass : TClass; AList : TStrings)');
17388: end;
17389:
17390: procedure SIRegister_ToolsUnit(CL: TPSPascalCompiler);
17391: begin
17392:   'MaxDataSet', 'LongInt').SetInt( 35);
17393:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17394:   SIRegister_TDBConnector(CL);
17395:   SIRegister_TDBBasicsTestSetup(CL);
17396:   SIRegister_TTestDataLink(CL);
17397:   'testValuesCount','Longint').SetInt( 25);
17398:   Procedure InitialiseDBConnector');
17399:   Procedure FreeDBConnector');
17400:   Function DateTimeToString( d : tdatetime) : string');
17401:   Function TStringToDate( d : String) : TDate');
17402: end;
17403:
17404: procedure SIRegister_fpcunit(CL: TPSPascalCompiler);
17405: begin
17406:   SIRegister_EAssertionFailedError(CL);
17407:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17408:   CL.AddTypeS('TRunMethod', 'Procedure');
17409:   CL.AddClass(CL.FindClass('TOBJECT'),'TTestResult');
17410:   SIRegister_TTest(CL);
17411:   SIRegister_TAssert(CL);
17412:   SIRegister_TTestFailure(CL);
17413:   SIRegister_ITestListener(CL);
17414:   SIRegister_TTestCase(CL);
17415:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17416:   SIRegister_TTestSuite(CL);
17417:   SIRegister_TTestResult(CL);
17418:   Function ComparisonMsg( const aExpected : string; const aActual : string) : string');
17419: end;
17420:
17421: procedure SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
17422: begin
17423:   TOBJECT','ETCPBuffer');
17424:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Integer;
17425:     +r; Head : Integer; Used : Integer; end');
17426:   'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
17427:   'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
17428:   'TCP_BUFFER_DEFAULTMAXSIZE','Longint').SetInt( ETHERNET_MTU_1GBT * 4);
17429:   'TCP_BUFFER_DEFAULTBUFSIZE','Longint').SetInt( ETHERNET_MTU_100MBIT * 4);
17430:   Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int;const TCPBufSize:Int);
17431:   Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer)');
17432:   Procedure TCPBufferPack( var TCPBuf : TTCPBuffer)');
17433:   Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer)');
17434:   Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer)');
17435:   Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer)');
17436:   Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer) : Pointer');
17437:   Procedure TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf : string; const Size : Integer)');
17438:   Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Integer');
17439:   Function TCPBufferPeek( var TCPBuf : TTCPBuffer; var Buf: string; const Size:Integer): Integer');
17440:   Function TCPBufferRemove( var TCPBuf : TTCPBuffer;var Buf: string; const Size:Integer): Integer');
17441:   Procedure TCPBufferClear( var TCPBuf : TTCPBuffer)');
17442:   Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer) : Integer');
17443:   Function TCPBufferUsed( const TCPBuf : TTCPBuffer) : Integer');

```

```

17444: Function TCPBufferEmpty( const TCPBuf : TTCPBuffer ) : Boolean';
17445: Function TCPBufferAvailable( const TCPBuf : TTCPBuffer ) : Integer';
17446: Function TCPBufferPtr( const TCPBuf : TTCPBuffer ) : Pointer';
17447: Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer );
17448: end;
17449:
17450: procedure SIRegister_Glut(CL: TPSPascalCompiler);
17451: begin
17452: //CL.AddTypeS('PInteger', '^Integer // will not work');
17453: //CL.AddTypeS('PPChar', '^PChar // will not work');
17454: CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3 );
17455: CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12 );
17456: CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0 );
17457: CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5 );
17458: CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6 );
17459: Procedure LoadGlut( const dll : String );
17460: Procedure FreeGlut();
17461: end;
17462:
17463: procedure SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
17464: begin
17465: CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17466: Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean ) : THandle';
17467: end;
17468:
17469: procedure SIRegister_SwitchLed(CL: TPSPascalCompiler);
17470: begin
17471: TLedColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black )';
17472: TTLedState', '( LedOn, LedOff, LedDisabled )';
17473: SIRegister_TSwitchLed(CL);
17474: //Procedure Register';
17475: end;
17476:
17477: procedure SIRegister_FileClass(CL: TPSPascalCompiler);
17478: begin
17479: CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17480: +'xisting, fdOpenAlways, fdTruncateExisting )');
17481: CL.AddTypeS('TFileAccess', '( faRead, faWrite, faReadWrite )');
17482: CL.AddTypeS('TFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17483: SIRegister_TCustomFile(CL);
17484: SIRegister_TIFile(CL);
17485: SIRegister_TMemoryFile(CL);
17486: SIRegister_TTextFileReader(CL);
17487: SIRegister_TTextFileWriter(CL);
17488: SIRegister_TFileMapping(CL);
17489: SIRegister_EFileError(CL);
17490: end;
17491:
17492: procedure SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
17493: begin
17494: CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17495: +' , ffaDirectory, ffaArchive, ffaAnyFile )');
17496: CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17497: SIRegister_TFileSearch(CL);
17498: end;
17499:
17500: procedure SIRegister_uColorFunctions(CL: TPSPascalCompiler);
17501: begin
17502: TRGBTType', 'record RedHex : string; GreenHex : string; BlueHex :'
17503: +' string; Red : integer; Green : integer; Blue : integer; end');
17504: Function FadeColor( aColor : Longint; aFade : integer ) : Tcolor';
17505: Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17506: end;
17507:
17508: procedure SIRegister_uSettings(CL: TPSPascalCompiler);
17509: begin
17510: Procedure SaveOscSettings());
17511: Procedure GetOscSettings();
17512: end;
17513:
17514: procedure SIRegister_cyDebug(CL: TPSPascalCompiler);
17515: begin
17516: TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer )';
17517: RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17518: 'FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17519: '64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17520: 'Cardinal; EnterCount : Integer; ExitCount : Integer; end';
17521: SIRegister_TcyDebug(CL);
17522: end;
17523:
17524: (*-----*)
17525: procedure SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
17526: begin
17527: TCopyFileResult', '( cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )';
17528: TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )';
17529: TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )';
17530: SIRegister_TdestinationOptions(CL);
17531: TProcCustomCopyFileEvent', 'Procedure ( Sender : TObject; var CopyFileResult : TCopyFileResult )';
17532: TProcOnCopyFileProgressEvent', 'Procedure(Sender:TObject;FileBytes,
TransferredBytes:int64;PercentDone:Int64);
```

```

17533: TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String)');
17534: SIRegister_TcyCopyFiles(CL);
17535: Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
ResetAttr:boolean): TCopyFileResult');
17536: Function cyCopyFileEx(FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult');
17537: Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;' +
FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer');
17538: end;
17539: 
17540: 
17541: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17542: begin
17543:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17544:   SIRegister_TcyFileAttributes(CL);
17545:   SIRegister_TSearchRecInstance(CL);
17546:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17547:   TOptions', 'set of TOption');
17548:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17549:   TProcOnValidateFileEvent Procedure(Sender: TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
Accept:bool;
17550:   TProcOnValidateDirectoryEvent', 'Procedure(Sender:TObject;Directory:String;var Accept:boolean)');
17551:   SIRegister_TcyCustomSearchfiles(CL);
17552:   SIRegister_TcySearchFiles(CL);
17553:   Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean');
17554:   Function IscyFolder( aSRec : TSearchrec) : Boolean');
17555: end;
17556: 
17557: procedure SIRegister_jcontrolutils(CL: TPSPascalCompiler);
17558: begin
17559:   Function jCountChar( const s : string; ch : char) : integer');
17560:   Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings)');
17561:   Function jNormalizeDate(const Value: string; theValue: TDateTime;const theFormat:string): string');
17562:   Function jNormalizeTime(const Value: string; theValue: TTime;const theFormat : string) : string');
17563:   Function jNormalizeDateTime(const Value:string;theValue:TDateTime;const theFormat:string):string');
17564:   Function jNormalizeDateSeparator( const s : string) : string');
17565:   Function jIsValidDateString( const Value : string) : boolean');
17566:   Function jIsValidTimeString( const Value : string) : boolean');
17567:   Function jIsValidDateTimeString( const Value : string) : boolean');
17568: end;
17569: 
17570: procedure SIRegister_kcMapView(CL: TPSPascalCompiler);
17571: begin
17572:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMapView');
17573:   CL.AddTypeS('TMapViewSource', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo' +
'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik' +
', msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual' +
'EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa' +
'hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17574:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17575:   TRealArea', 'record top : Extended; left : Extended; bottom : Extended; right : Extended; end');
17576:   TIntPoint', 'record X : Int64; Y : Int64; end');
17577:   TkcRealPoint', 'record X : Extended; Y : Extended; end');
17578:   TOnBeforeDownloadEvent', 'Procedure ( Url : string; str : TStream; var CanHandle : Boolean)');
17579:   TOnAfterDownloadEvent', 'Procedure ( Url : string; str : TStream)');
17580:   SIRegister_TCustomeDownloadEngine(CL);
17581:   SIRegister_TCustomGeolocationEngine(CL);
17582:   SIRegister_TMapView(CL);
17583: end;
17584: 
17585: procedure SIRegister_cparserutils(CL: TPSPascalCompiler);
17586: begin
17587:   (*Function isFunc( name : TNamePart ) : Boolean');*)
17588:   Function isUnnamedFunc( name : TNamepart ) : Boolean');
17589:   Function isPtrToFunc( name : TNamePart ) : Boolean');
17590:   Function isFuncRetFuncPtr( name : TNamePart ) : Boolean');
17591:   Function isPtrToFuncRetFuncPtr( name : TNamePart ) : Boolean');
17592:   Function GetFuncParam( name : TNamePart ) : TNamePart');
17593:   Function isArray( name : TNamePart ) : Boolean');
17594:   Function GetArrayPart( name : TNamePart ) : TNamePart');
17595:   Function GetIdFromPart( name : TNamePart ) : AnsiString');
17596:   Function GetIdPart( name : TNamePart ) : TNamePart');
17597:   Function isNamePartPtrToFunc( part : TNamePart ) : Boolean');
17598:   Function isAnyBlock( part : TNamePart ) : Boolean');*/
17599:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17600:   SIRegister_TLineBreaker(CL);
17601:   CL.AddTypeS('TNameKind', 'Integer');
17602:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNamePart');
17603:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17604:   Function SphericalMod( X : Extended) : Extended');
17605:   Function cSign( Value : Extended) : Extended');
17606:   Function LimitFloat( const eValue, eMin, eMax : Extended) : Extended');
17607:   Function AngleToRadians( iAngle : Extended) : Extended');
17608:   Function RadiansToAngle( eRad : Extended) : Extended');
17609:   Function Cross180( iLong : Double) : Boolean');
17610:   Function Mod180( Value : integer) : Integer');
17611:   Function Mod180Float( Value : Extended) : Extended');
17612:   Function MulDivFloat( a, b, d : Extended) : Extended');
17613:   Function LongDiff( iLong1, iLong2 : Double) : Double');

```

```

17618: Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap)');
17619: Function Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap');
17620: Function FixFilePath( const Inpath, CheckPath : string) : string');
17621: Function UnFixFilePath( const Inpath, CheckPath : string) : string');
17622: Procedure FillStringList( sl : TStringList; const aText : string)');
17623: end;
17624:
17625: procedure SIRegister_LedNumber(CL: TPSPPascalCompiler);
17626: begin
17627:   TLedSegmentSize', 'Integer');
17628:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised )');
17629:   SIRegister_TCustomLEDNumber(CL);
17630:   SIRegister_TLEDNumber(CL);
17631: end;
17632:
17633: procedure SIRegister_StStrL(CL: TPSPPascalCompiler);
17634: begin
17635:   CL.AddTypeS('LStrRec', 'record AllocSize : Longint; RefCount : Longint; Length : Longint; end');
17636:   CL.AddTypes('AnsiChar', 'Char');
17637:   CL.AddTypeS('BTable', 'array[0..255] of Byte'); //!!!
17638:   Function HexBL( B : Byte) : AnsiString';
17639:   Function HexWL( W : Word) : AnsiString';
17640:   Function HexLL( L : LongInt) : AnsiString';
17641:   Function HexPTRL( P : __Pointer) : AnsiString';
17642:   Function BinaryBL( B : Byte) : AnsiString';
17643:   Function BinaryWL( W : Word) : AnsiString';
17644:   Function BinaryLL( L : LongInt) : AnsiString';
17645:   Function OctalBL( B : Byte) : AnsiString';
17646:   Function OctalWL( W : Word) : AnsiString';
17647:   Function OctalLL( L : LongInt) : AnsiString';
17648:   Function Str2Int16L( const S : AnsiString; var I : SmallInt) : Boolean';
17649:   Function Str2WordL( const S : AnsiString; var I : Word) : Boolean';
17650:   Function Str2LongL( const S : AnsiString; var I : LongInt) : Boolean';
17651:   Function Str2RealL( const S : AnsiString; var R : Double) : Boolean';
17652:   Function Str2RealL( const S : AnsiString; var R : Real) : Boolean';
17653:   Function Str2ExtL( const S : AnsiString; var R : Extended) : Boolean';
17654:   Function Long2StrL( L : LongInt) : AnsiString';
17655:   Function Real2StrL( R : Double; Width : Byte; Places : ShortInt) : AnsiString';
17656:   Function Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : AnsiString';
17657:   Function ValPrepL( const S : AnsiString) : AnsiString';
17658:   Function CharStrL( C : Char; Len : Cardinal) : AnsiString';
17659:   Function PadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString';
17660:   Function PadLL( const S : AnsiString; Len : Cardinal) : AnsiString';
17661:   Function LeftPadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString';
17662:   Function LeftPadL( const S : AnsiString; Len : Cardinal) : AnsiString';
17663:   Function TrimLeadL( const S : AnsiString) : AnsiString';
17664:   Function TrimTrailL( const S : AnsiString) : AnsiString';
17665:   Function TrimL( const S : AnsiString) : AnsiString';
17666:   Function TrimSpacesL( const S : AnsiString) : AnsiString';
17667:   Function CenterChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString';
17668:   Function CenterL( const S : AnsiString; Len : Cardinal) : AnsiString';
17669:   Function EntabL( const S : AnsiString; TabSize : Byte) : AnsiString';
17670:   Function DetabL( const S : AnsiString; TabSize : Byte) : AnsiString';
17671:   Function ScrambleL( const S, Key : AnsiString) : AnsiString';
17672:   Function SubstituteL( const S, FromStr,ToStr : AnsiString) : AnsiString';
17673:   Function FilterL( const S, Filters : AnsiString) : AnsiString';
17674:   Function CharExistsL( const S : AnsiString; C : AnsiChar) : Boolean';
17675:   Function CharCountL( const S : AnsiString; C : AnsiChar) : Cardinal';
17676:   Function WordCountL( const S, WordDelims : AnsiString) : Cardinal';
17677:   Function WordPositionL( N : Cardinal; const S, WordDelims : AnsiString; var Pos : Cardinal) : Boolean';
17678:   Function ExtractWordL( N : Cardinal; const S, WordDelims : AnsiString) : AnsiString';
17679:   Function AsciiCountL( const S, WordDelims : AnsiString; Quote : AnsiChar) : Cardinal';
17680:   Function AsciiPositionL(N: Cardinal;const S,WordDelims:AnsiString;Quote:AnsiChar;var Pos:Cardinal):Bool;
17681:   Function ExtractAsciiL( N : Cardinal; const S, WordDelims : AnsiString; Quote : AnsiChar) : AnsiString';
17682:   Procedure WordWrapL(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17683:   Procedure WordWrap(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17684:   Function CompStringL( const S1, S2 : AnsiString) : Integer';
17685:   Function CompUCStringL( const S1, S2 : AnsiString) : Integer';
17686:   Function SoundexL( const S : AnsiString) : AnsiString';
17687:   Function MakeLetterSetL( const S : AnsiString) : Longint';
17688:   Procedure BMMakeTableL( const MatchString : AnsiString; var BT : BTable)');
17689:   Function BMSearchL(var Buffer,BufLength:Cardinal;var BT:BTable;const MatchString:AnsiString;var
Pos:Card):Bool;
17690:   Function BMSearchUCL( var Buffer, BufLength : Cardinal; var BT : BTable; const MatchString : AnsiString;
var Pos : Cardinal) : Boolean';
17691:   Function DefaultExtensionL( const Name, Ext : AnsiString) : AnsiString';
17692:   Function ForceExtensionL( const Name, Ext : AnsiString) : AnsiString';
17693:   Function JustFilenameL( const PathName : AnsiString) : AnsiString';
17694:   Function JustNameL( const PathName : AnsiString) : AnsiString';
17695:   Function JustExtensionL( const Name : AnsiString) : AnsiString';
17696:   Function JustPathnameL( const PathName : AnsiString) : AnsiString';
17697:   Function AddBackSlashL( const DirName : AnsiString) : AnsiString';
17698:   Function CleanPathNameL( const PathName : AnsiString) : AnsiString';
17699:   Function HasExtensionL( const Name : AnsiString; var DotPos : Cardinal) : Boolean';
17700:   Function CommaizeL( L : Longint) : AnsiString';
17701:   Function CommaizeChL( L : Longint; Ch : AnsiChar) : AnsiString';
17702:   Function FloatFormL(const Mask:AnsiString;R:TstFloat;const LtCurr:AnsiString;Sep,
DecPt:Char):AnsiString;
17703:   Function LongIntFormL(const Mask:AnsiString;L:LongInt;const LtCurr,
RtCurr:AnsiString;Sep:Char):AnsiString';

```

```

17704: Function StrChPosL( const P : AnsiString; C : AnsiChar; var Pos : Cardinal) : Boolean';
17705: Function StrStPosL( const P, S : AnsiString; var Pos : Cardinal) : Boolean');
17706: Function StrStCopyL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17707: Function StrChInsertL( const S : AnsiString; C : AnsiChar; Pos : Cardinal) : AnsiString');
17708: Function StrStInsertL( const S1, S2 : AnsiString; Pos : Cardinal) : AnsiString');
17709: Function StrChDeleteL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17710: Function StrStDeleteL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17711: Function ContainsOnlyL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17712: Function ContainsOtherThanL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17713: Function CopyLeftL( const S : AnsiString; Len : Cardinal) : AnsiString');
17714: Function CopyMidL( const S : AnsiString; First, Len : Cardinal) : AnsiString');
17715: Function CopyRightL( const S : AnsiString; First : Cardinal) : AnsiString');
17716: Function CopyRightAbsL( const S : AnsiString; NumChars : Cardinal) : AnsiString');
17717: Function CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Cardinal;var
SubString:AString):Bool;
17718: Function CopyFromToWordL(const S,WordDelims,Word1,Word2:AnsiString;N1,N2:Cardinal;var
SubString:AnsiString):Bool;
17719: Function CopyWithinL( const S, Delimiter : AnsiString; Strip : Boolean) : AnsiString');
17720: Function DeleteFromNthWordL( const S, WordDelims : AnsiString; const AWord : AnsiString; N : Cardinal;
var SubString : AnsiString) : Boolean');
17721: Function DeleteFromToWordL( const S, WordDelims, Word1, Word2 : AnsiString; N1, N2 : Cardinal; var
SubString : AnsiString) : Boolean');
17722: Function DeleteWithinL( const S, Delimiter : AnsiString) : AnsiString');
17723: Function ExtractTokensL(const S,
Delims:AnsiString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Cardinal;
17724: Function IsChAlphaL( C : AnsiChar ) : Boolean');
17725: Function IsChNumericL( C : AnsiChar; const Numbers : AnsiString ) : Boolean');
17726: Function IsChAlphaNumericL( C : AnsiChar; const Numbers : AnsiString ) : Boolean');
17727: Function IsStrAlphaL( const S : AnsiString ) : Boolean');
17728: Function IsStrNumericL( const S, Numbers : AnsiString ) : Boolean');
17729: Function IsStrAlphaNumericL( const S, Numbers : AnsiString ) : Boolean');
17730: Function KeepCharsL( const S, Chars : AnsiString ) : AnsiString');
17731: Function LastWordL( const S, WordDelims, AWord : AnsiString; var Position : Cardinal ) : Boolean');
17732: Function LastWordAbsL( const S, WordDelims : AnsiString; var Position : Cardinal ) : Boolean');
17733: Function LastStringL( const S, AString : AnsiString; var Position : Cardinal ) : Boolean');
17734: Function LeftTrimCharsL( const S, Chars : AnsiString ) : AnsiString');
17735: Function ReplaceWordL(const S, WordDelims,OldWord,NewWord:AnsiString;N:Card;var
Replacements:Card):AnsiString;
17736: Function ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:AnsiString;var
Replacements:Cardinal):AnsiString');
17737: Function ReplaceStringL(const S,OldString,NewString:AnsiString;N:Cardinal;var
Replacements:Cardinal):AnsiString;
17738: Function ReplaceStringAllL(const S,OldString,NewString:AnsiString; var Replacements:Cardinal):AnsiString;
17739: Function RepeatStringL(const RepeatString:AnsiString;var Repetitions:Cardinal;MaxLen:Cardinal):AnsiString;
17740: Function RightTrimCharsL( const S, Chars : AnsiString ) : AnsiString');
17741: Function StrWithinL( const S, SearchStr : string; Start : Cardinal; var Position : Cardinal ) : boolean');
17742: Function TrimCharsL( const S, Chars : AnsiString ) : AnsiString');
17743: Function WordPosL(const S,WordDelims,AWord: AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17744: Function WordPos(const S,WordDelims,AWord:AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17745: end;
17746:
17747: procedure SIRegister_pwnative_out(CL: TPSPascalCompiler);
17748: begin
17749: CL.AddConstantN('STDIN','LongInt').SetInt( 0 );
17750: ('STDOUT','LongInt').SetInt( 1 );
17751: ('STDERR','LongInt').SetInt( 2 );
17752: Procedure NativeWrite( s : astr );';
17753: Procedure NativeWrite1( PString : PChar );';
17754: Procedure NativeWrite2( Buffer : PChar; NumChars : Cardinal );';
17755: Procedure NativeWriteLn( s : astr );';
17756: Procedure NativeWriteLn1();';
17757: end;
17758:
17759: procedure SIRegister_synwrapl(CL: TPSPascalCompiler);
17760: begin
17761: CL.AddTypeS(TSynwInfo', 'record Err : byte; UrlHtml : ansistring; ErrResponse
17762: : integer; UltimateURL : ansistring; Headers : ansistring; end');
17763: CL.AddTypeS('TUrlInfo', 'record Err : byte; UltimateURL : string; end');
17764: Function GetHttpFile( const Url, UserAgent, Outfile : string; verbose : boolean): TSynwInfo;');
17765: Function GetHttpFile1( const Url, UserAgent, Outfile : string ) : TSynwInfo;');
17766: Function GetHttpFile2( const Url, Outfile : string ) : TSynwInfo;');
17767: Function GetHttpFile3( const Url, outfile : string; verbose : boolean ) : TSynwInfo;');
17768: Function GetHtm( const Url : string ) : string;');
17769: Function GetHtm1( const Url, UserAgent : string ) : string;');
17770: Function GetUrl( const Url : string; verbose : boolean ) : TSynwInfo;');
17771: Function GetUrl1( const Url, useragent : string ) : TSynwInfo;');
17772: Function GetUrl2( const Url : string ) : TSynwInfo;');
17773: Function GetUrl3( const Url : string; const http : THTTPSSend; verbose : boolean): TUrlInfo;');
17774: Function GetUrl4( const Url : string; const http : THTTPSSend ) : TUrlInfo;');
17775: Procedure StrToStream( s : String; strm : TMemoryStream );
17776: Function StrLoadStream( strm : TStream ) : String');
17777: end;
17778:
17779: procedure SIRegister_HTMLUtil(CL: TPSPascalCompiler);
17780: begin
17781: Function GetVal( const tag, attribname_ci : string ) : string');
17782: Function GetTagName( const Tag : string ) : string');
17783: Function GetUpTagName( const tag : string ) : string');
17784: Function GetNameValPair( const tag, attribname_ci : string ) : string');

```

```

17785: Function GetValFromNameVal( const namevalpair : string ) : string';
17786: Function GetNameValPair_cs( const tag, attribname : string ) : string';
17787: Function GetVal_JAMES( const tag, attribname_ci : string ) : string';
17788: Function GetNameValPair_JAMES( const tag, attribname_ci : string ) : string');
17789: Function CopyBuffer( StartIndex : PChar; Len : integer ) : string');
17790: Function Ucase( s : string ) : string');
17791: end;
17792:
17793: procedure SIRegister_pwmain(CL: TPSCompiler);
17794: begin
17795:   CL.AddConstantN('FUTURE_COOKIE','String').SetString( 'Mon, 01 Dec 2099 12:00:00 GMT');
17796:   EXPIRED_COOKIE'.SetString( 'Mon, 01 Jan 2001 12:00:00 GMT');
17797:   'SECURE_OFF', LongInt').SetInt( 0 );
17798:   'SECURE_ON', LongInt').SetInt( 2 );
17799:   'SECURE_FILTER', LongInt').SetInt( 3 );
17800:   THandle or DWord!
17801: //  astr = ansistring;
17802: CL.AddTypeS('pastr', 'ansistring');
17803: CL.AddTypeS('TFilterFunc', 'function(const s: pastr): pastr;');
17804: uses pwinit at begin
17805: //type TFilterFunc = function(const s: astr): astr;
17806: Demo: ..\maxbox3\examples2\519_powlts.txt
17807:
17808: //CL.AddConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false );
17809: //CL.AddConstantN('CASE_IGNORE','Boolean')BoolToStr( false );
17810: Procedure pwInit();
17811: Procedure Offreadln();
17812: Function Lcase( const s : pastr ) : pastr';
17813: Function Ucase( const s : pastr ) : pastr';
17814: Function CountPostVars : longword';
17815: Function GetPostVar( const name : pastr ) : pastr';
17816: Function GetPostVarL( const name : pastr; filter : TFilterFunc ) : pastr';
17817: Function GetPostVar_S( const name : pastr; Security : integer ) : pastr';
17818: Function GetPostVar_SF( const name : pastr; Security : integer ) : pastr';
17819: Function GetPostVarAsFloat( const name : pastr ) : double';
17820: Function GetPostVarAsInt( const name : pastr ) : longint';
17821: Function GetPostVar_SafeHTML( const name : pastr ) : pastr';
17822: Function FetchPostVarName( idx : longword ) : pastr';
17823: Function FetchPostVarVal( idx : longword ) : pastr';
17824: Function FetchPostVarValL( idx : longword; filter : TFilterFunc ) : pastr';
17825: Function FetchPostVarName_S( idx : longword; Security : integer ) : pastr';
17826: Function FetchPostVarVal_S( idx : longword; Security : integer ) : pastr';
17827: Function IsPostVar( const name : pastr ) : boolean';
17828: Function CountAny : longword';
17829: Function GetAny( const name : pastr ) : pastr';
17830: Function GetAnyL( const name : pastr; filter : TFilterFunc ) : pastr';
17831: Function GetAny_S( const name : pastr; Security : integer ) : pastr';
17832: Function GetAnyAsFloat( const name : pastr ) : double';
17833: Function GetAnyAsInt( const name : pastr ) : longint';
17834: Function IsAny( const name : pastr ) : byte';
17835: Function CountCookies : longword';
17836: Function FetchCookieName( idx : longword ) : pastr';
17837: Function FetchCookieVal( idx : longword ) : pastr';
17838: Function FetchCookieValL( idx : longword; filter : TFilterFunc ) : pastr';
17839: Function GetCookie( const name : pastr ) : pastr';
17840: Function GetCookieL( const name : pastr; filter : TFilterFunc ) : pastr';
17841: Function GetCookieAsFloat( const name : pastr ) : double';
17842: Function GetCookieAsInt( const name : pastr ) : longint';
17843: Function IsCookie( const name : pastr ) : boolean';
17844: Function SetCookie( const name, value : pastr ) : boolean';
17845: Function SetCookieAsFloat( const name : pastr; value : double ) : boolean';
17846: Function SetCookieAsInt( const name : pastr; value : longint ) : boolean';
17847: Function SetCookieEx( const name, value, path, domain, expiry : pastr ) : boolean';
17848: Function SetCookieAsFloatEx( const name:pastr;value : double; const path,domain,expiry:pastr ):bool;
17849: Function SetCookieAsIntEx( const name:pastr;value : longint; const path,domain,expiry:pastr ):bool;
17850: Function UnsetCookie( const name : pastr ) : boolean';
17851: Function UnsetCookieEx( const name, path, domain : pastr ) : boolean';
17852: Function FilterHtml( const input : pastr ) : pastr';
17853: Function FilterHtml_S( const input : pastr; security : integer ) : pastr';
17854: Function TrimBadChars( const input : pastr ) : pastr';
17855: Function TrimBadFile( const input : pastr ) : pastr';
17856: Function TrimBadDir( const input : pastr ) : pastr';
17857: Function TrimBad_S( const input : pastr; security : integer ) : pastr';
17858: Function CountHeaders : longword';
17859: Function FetchHeaderName( idx : longword ) : pastr';
17860: Function FetchHeaderVal( idx : longword ) : pastr';
17861: Function GetHeader( const name : pastr ) : pastr';
17862: Function IsHeader( const name : pastr ) : boolean';
17863: Function SetHeader( const name, value : pastr ) : boolean';
17864: Function UnsetHeader( const name : pastr ) : boolean';
17865: Function PutHeader( const header : pastr ) : boolean';
17866: Procedure OutL( const s : pastr );
17867: Procedure OutLn( const s : pastr );
17868: Procedure OutA( args : array of const );
17869: Procedure OutF( const s : pastr );
17870: Procedure OutLnF( const s : pastr );
17871: Procedure OutFF( const s : pastr );
17872: Procedure OutF_FI( const s : pastr; HTMLFilter : boolean );
17873: Procedure OutLnFF( const s : pastr );

```

```

17874: Procedure OutLnF_FI( const s : pastr; HTMLFilter : boolean );
17875: Function FileOut( const fname : pastr ) : word';
17876: Function ResourceOut( const fname : pastr ) : word';
17877: Procedure BufferOut( const buff, len : LongWord );
17878: Function TemplateOut( const fname : pastr; HtmlFilter : boolean ) : word';
17879: Function TemplateOut1( const fname : pastr ) : word';
17880: Function TemplateOut2( const fname : pastr; filter : TFilterFunc ) : word';
17881: Function TemplateOut3( const fname : pastr; HtmlFilter : boolean ) : word';
17882: Function TemplateRaw( const fname : pastr ) : word';
17883: Function Fmt( const s : pastr ) : pastr');
17884: Function Fmt1( const s : pastr; filter : TFilterFunc ) : pastr');
17885: Function Fmt2( const s : pastr ) : pastr');
17886: Function Fmt_SF( const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecurity:int ):pastr;
17887: Function Fmt_SFL( const s:pastr; HTMLFilter:boolean; FilterSecurity, TrimSecurity : integer ) : pastr');
17888: Function CountRtiVars : longword';
17889: Function FetchRtiName( idx : longword ) : pastr';
17890: Function FetchRtiVal( idx : longword ) : pastr';
17891: Function GetRti( const name : pastr ) : pastr';
17892: Function GetRtiAsFloat( const name : pastr ) : double';
17893: Function GetRtiAsInt( const name : pastr ) : longint';
17894: Function IsRti( const name : pastr ) : boolean';
17895: Procedure SetRTI( const name, value : pastr );
17896: Function FetchUpfileName( idx : longword ) : pastr';
17897: Function GetUpfileName( const name : pastr ) : pastr';
17898: Function GetUpfileSize( const name : pastr ) : longint';
17899: Function GetUpFileType( const name : pastr ) : pastr';
17900: Function CountUpfiles : longword';
17901: Function IsUpfile( const name : pastr ) : boolean';
17902: Function SaveUpfile( const name, fname : pastr ) : boolean';
17903: Function CountVars : longword';
17904: Function FetchVarName( idx : longword ) : pastr';
17905: Function FetchVarVal( idx : longword ) : pastr';
17906: Function FetchVarVal1( idx : longword; filter : TFilterFunc ) : pastr');
17907: Function GetVar( const name : pastr ) : pastr');
17908: Function GetVar1( const name : pastr; filter : TFilterFunc ) : pastr');
17909: Function GetVar_S( const name : pastr; security : integer ) : pastr');
17910: Function GetVarAsFloat( const name : pastr ) : double';
17911: Function GetVarAsInt( const name : pastr ) : longint';
17912: Procedure SetVar( const name, value : pastr );
17913: Procedure SetVarAsFloat( const name : pastr; value : double );
17914: Procedure SetVarAsInt( const name : pastr; value : longint );
17915: Function IsVar( const name : pastr ) : byte';
17916: Procedure UnsetVar( const name : pastr );
17917: Function LineEndToBR( const s : pastr ) : pastr';
17918: Function RandomStr( len : longint ) : pastr';
17919: Function XorCrypt( const s : pastr; key : byte ) : pastr';
17920: Function CountCfgVars : longword';
17921: Function FetchCfgVarName( idx : longword ) : pastr';
17922: Function FetchCfgVarVal( idx : longword ) : pastr';
17923: Function IsCfgVar( const name : pastr ) : boolean';
17924: Function SetCfgVar( const name, value : pastr ) : boolean';
17925: Function GetCfgVar( const name : pastr ) : pastr';
17926: Procedure ThrowErr( const s : pastr );
17927: Procedure ThrowWarn( const s : pastr );
17928: Procedure ErrWithHeader( const s : pastr );
17929: CL.AddTypeS('TWebVar', 'record name : pastr; value : pastr; end');
17930: CL.AddTypeS('TWebVars', 'array of TWebVar');
17931: Function iUpdateWebVar(var webv : TWebVars; const name, value:pastr; upcased:boolean):boolean';
17932: Function iAddWebCfgVar( const name, value : pastr ) : boolean';
17933: Procedure iAddWebVar( var webv : TWebVars; const name, value : pastr );
17934: Procedure iSetRTI( const name, value : pastr );
17935: Function iCustomSessUnitSet : boolean';
17936: Function iCustomCfgUnitSet : boolean';
17937: end;
17938:
17939: procedure SIRегистre_W32VersionInfo(CL: TPSPascalCompiler);
17940: begin
17941:   SIRегистre_TProjectVersionInfo(CL);
17942:   Function MSLanguageToHex( const s : string ) : string';
17943:   Function MSHexToLanguage( const s : string ) : string';
17944:   Function MSCharacterSetToHex( const s : string ) : string';
17945:   Function MSHexToCharacterSet( const s : string ) : string';
17946:   Function MSLanguages : TStringList';
17947:   Function MSHexLanguages : TStringList';
17948:   Function MSCharacterSets : TStringList';
17949:   Function MSHexCharacterSets : TStringList';
17950: end;
17951:
17952: procedure SIRегистre_IpUtils(CL: TPSPascalCompiler);
17953: begin
17954:   TIpHandle', 'Cardinal');
17955:   TIpMD5StateArray', 'array[0..3] of DWORD';
17956:   CL.AddTypeS('TIpMD5CountArray', 'array[0..1] of DWORD');
17957:   TIpMD5ByteBuf', 'array[0..63] of Byte';
17958:   TIpMD5LongBuf', 'array[0..15] of DWORD';
17959:   TIpMD5Digest', 'array[0..15] of Byte';
17960:   TIpLineTerminator', '( ltNone, ltCR, ltLF, ltCRLF, ltOther )';
17961:   TIpMD5Context', 'record State : TIpMD5StateArray; Count : TIpMD5';
17962:     +'CountArray; ByteBuf : TIpMD5ByteBuf; end');

```

```

17963: CL.AddClassN(CL.FindClass('TOBJECT'), 'EIpBaseException');
17964: CL.AddClassN(CL.FindClass('TOBJECT'), 'EIpAccessException');
17965: CL.AddClassN(CL.FindClass('TOBJECT'), 'EIpHtmlException');
17966: SIRegister_TIpBaseAccess(CL);
17967: SIRegister_TIpBasePersistent(CL);
17968: //TIPComponentClass', 'class of TIpBaseComponent');
17969: SIRegister_TIpBaseComponent(CL);
17970: SIRegister_TIpBaseWinControl(CL);
17971: Function InClassA( Addr : LongInt ) : Boolean';
17972: Function InClassB( Addr : LongInt ) : Boolean';
17973: Function InClassC( Addr : LongInt ) : Boolean';
17974: Function InClassD( Addr : LongInt ) : Boolean';
17975: Function InMulticast( Addr : LongInt ) : Boolean';
17976: Function IpCharCount( const Buffer, BufSize : DWORD; C : AnsiChar ) : DWORD';
17977: Function IpCompStruct( const S1, S2, Size : Cardinal ) : Integer';
17978: Function IpMaxInt( A, B : Integer ) : Integer';
17979: Function IpMinInt( A, B : Integer ) : Integer';
17980: Procedure IpSafeFree( var Obj: TObject );
17981: Function IpShortVersion : string';
17982: Function IpInternetSumPrim( var Data, DataSize, CurCrc : DWORD ) : DWORD';
17983: Function IpInternetSumOfStream( Stream : TStream; CurCrc : DWORD ) : DWORD';
17984: Function IpInternetSumOfFile( const FileName : string ) : DWORD';
17985: Function MD5SumOfFile( const FileName : string ) : string';
17986: Function MD5SumOfStream( Stream : TStream ) : string';
17987: Function MD5SumOfStreamDigest( Stream : TStream ) : TIpMD5Digest';
17988: Function MD5SumOfString( const S : string ) : string';
17989: Function MD5SumOfStringDigest( const S : string ) : TIpMD5Digest';
17990: Function SafeYield : LongInt';
17991: Function AllTrimSpaces( Strng : string ) : string';
17992: Function IpCharPos( C : AnsiChar; const S : string ) : Integer';
17993: Function CharPosIdx( C : AnsiChar; const S : string; Idx : Integer ) : Integer';
17994: Function NthCharPos( C : AnsiChar; const S : string; Nth : Integer ) : Integer';
17995: Function RCharPos( C : AnsiChar; const S : string ) : Integer';
17996: Function RCharPosIdx( C : AnsiChar; const S : string; Idx : Integer ) : Integer';
17997: Function RNthCharPos( C : AnsiChar; const S : string; Nth : Integer ) : Integer';
17998: Function IpRPos( const Substr : string; const S : string ) : Integer';
17999: Function IpPosIdx( const Substr, S : string; Idx : Integer ) : Integer';
18000: ACharSet', 'set of AnsiChar');
18001: TIpAddrRec', 'record Scheme : string; UserName : string; Password string; Authority : string;
18002: Port : string; Path : string; Fragment : string; Query : string; QueryDelim : AnsiChar; end');
18003: Procedure Initialize( var AddrRec : TIpAddrRec );
18004: Procedure Finalize( var AddrRec : TIpAddrRec );
18005: Function ExtractEntityName( const NamePath : string ) : string';
18006: Function ExtractEntityPath( const NamePath : string ) : string';
18007: Function IpParseURL( const URL : string; var Rslt : TIpAddrRec ) : Boolean';
18008: Function BuildURL( const OldURL, NewURL : string ) : string';
18009: Function PutEscapes( const S : string; EscapeSet : ACharSet ) : string';
18010: Function RemoveEscapes( const S : string; EscapeSet : ACharSet ) : string';
18011: Procedure SplitParams( const Params : string; Dest : TStrings );
18012: Function NetToDOSPath( const PathStr : string ) : string';
18013: Function DOSToNetPath( const PathStr : string ) : string';
18014: Procedure SplitHttpResponse( const S : string; var V, MsgID, Msg : string );
18015: Procedure FieldFix( Fields : TStrings );
18016: Function AppendSlash( APath : string ) : string';
18017: Function RemoveSlash( APath : string ) : string';
18018: Function GetParentPath( const Path : string ) : string';
18019: Function GetLocalContent( const TheFileName : string ) : string';
18020: Function IPDirExists( Dir : string ) : Boolean';
18021: Function GetTemporaryFile( const Path : string ) : string';
18022: Function GetTemporaryPath : string';
18023: Function AppendBackSlash( APath : string ) : string';
18024: Function IpRemoveBackSlash( APath : string ) : string';
18025: Function INetDateToStrToDate( const DateStr : string ) : TDateTime';
18026: Function DateToString( Date : TDateTime ) : string';
18027: Function IpTimeZoneBias : Integer';
18028: Procedure SplitCookieFields( const Data : string; Fields : TStrings );
18029: end;
18030:
18031: procedure SIRegister_LrtPoTools(CL: TPSPascalCompiler);
18032: begin
18033:   CL.AddTypeS('TPOStyle', '( postStandard, postPropName, postFull )');
18034:   Procedure Lrt2Po( const LRTfile : string; PostStyle : TPOStyle );
18035:   Procedure CombinePoFiles( SL : TStrings; const FName : string );
18036:   end;
18037:
18038: procedure SIRegister_GPS(CL: TPSPascalCompiler);
18039: begin
18040:   CL.AddConstantN('MAX_SATS', 'LongInt').SetInt( 12 );
18041:   GPSMSG_START', 'String').SetString( '$' );
18042:   GPSMSG_STOP', 'String').SetString( '*' );
18043:   SEC_BETWEEN_SEG', 'LongInt').SetInt( 5 );
18044:   CL.AddTypeS('TSatellite', 'record Identification : Shortint; Elevation : Shor
18045:     +'tint; Azimut : Smallint; SignLevel : Smallint; end');
18046:   CL.AddTypeS('TSatellites', 'array[1..12] of TSatellite');
18047:   //TSatellites = array[1..MAX_SATS] of TSatellite;
18048:   TGPSSatEvent', 'Procedure ( Sender : TObject; NbSat, NbSatUse: Shortint; Sats : TSatellites )');
18049:   TGPSDatas', 'record Latitude : Double; Longitude : Double; Heigh
18050:     tAboveSea : Double; Speed : Double; UTCTime : TDateTime; Valid : Boolean;
18051:     NbrSats : Shortint; NbrSatsUsed : Shortint; Course : Double; end');

```

```

18052: CL.AddTypeS('TGPSDatasEvent', 'Procedure ( Sender : TObject; GPSDatas : TGPSDatas)');
18053: CL.AddTypeS('TMsgGP', '( msgGP, msgGPGGA, msgGPGLL, msgGPGSV, msgGPRMA, msgGPRMC, msgGPZDA )');
18054: CL.AddTypeS('TSpeedUnit', '( suKilometre, suMile, suNauticalMile )');
18055: SIRegister_TGPSLink(CL);
18056: SIRegister_TCustomGPS(CL);
18057: SIRegister_TGPS(CL);
18058: SIRegister_TGPSSoGPX(CL);
18059: SIRegister_TGPSSpeed(CL);
18060: SIRegister_TGPSSatellitesPosition(CL);
18061: SIRegister_TGPSSatellitesReception(CL);
18062: SIRegister_TGPSCompass(CL);
18063: //Procedure Register( );
18064: Function IndexMsgGP( StrMsgGP : String ) : TMsgGP';
18065: Function StrCoordToAngle( Point : Char; Angle : String ) : Double';
18066: Function StrTimeToTime( const Time : String ) : TDateTime';
18067: Function StrToInteger( const Str : String ) : Integer';
18068: Function StrToReal( const Str : String ) : Extended';
18069: Function GPSRotatePoint( Angle : Double; Ct, Pt : TPoint ) : TPoint';
18070: Procedure LoadRessource( RessourceName : String; ImageList : TImageList );
18071: end;
18072:
18073: procedure SIRegister_NMEA(CL: TPSPascalCompiler);
18074: begin
18075: NMEADataArray', 'array of string');
18076: Procedure TrimNMEA( var S : string );
18077: Procedure ExpandNMEA( var S : string );
18078: Function ParseNMEA( S : string ) : NMEADataArray';
18079: Function ChkValidNMEA( S : string ) : Boolean';
18080: Function IdNMEA( S : string ) : string';
18081: Function ChkSumNMEA( const S : string ) : string';
18082: Function PosInDeg( const PosStr : string ) : Double';
18083: Function DateNMEA( const StrD, StrT : string ) : TDateTime';
18084: Function SysClockSet( const StrD, StrT : string ) : Boolean';
18085: function Ticks2Secs(Ticks : LongInt) : LongInt';
18086: function Secs2Ticks(Secs : LongInt) : LongInt';
18087: function MSecs2Ticks(MSecs : LongInt) : LongInt';
18088: end;
18089:
18090: procedure SIRegister_SortUtils(CL: TPSPascalCompiler);
18091: begin
18092: CL.AddTypeS('SortType1', 'Byte');
18093: CL.AddTypeS('SortType2', 'Double');
18094: CL.AddTypeS('SortType3', 'DWord');
18095: //CL.AddTypeS('PDWordArray', '^DWordArray // will not work');
18096: CL.AddTypeS('TDataRecord4', 'record Value : Integer; Data : Integer; end');
18097: Function('Procedure QuickSort( var List : array of SortType1; Min, Max : Integer)');
18098: Procedure QuickSortDWord( var List : array of SortType3; Min, Max : Integer );
18099: Procedure QuickSortDataRecord4( var List : array of TDataRecord4; Count : Integer );
18100: Procedure HeapSort( var List : array of SortType1; Count : DWord; FirstNeeded : DWord );
18101: Function QuickSelect( var List : array of SortType1; Min, Max, Wanted : Integer ) : SortType1';
18102: Function QuickSelectDouble( var List : array of SortType2; Min, Max, Wanted : Integer ) : SortType2';
18103: Function QuickSelectDWord( var List : array of SortType3; Min, Max, Wanted : Integer ) : SortType3';
18104: end;
18105:
18106: procedure SIRegister_BitmapConversion(CL: TPSPascalCompiler);
18107: begin
18108: // TMatrix3x3 = array[1..3,1..3] of Double;
18109: // TMatrix4x4 = array[1..4,1..4] of Double;
18110: CL.AddTypeS('TMatrix3x31', 'array[1..3] of Double');
18111: CL.AddTypeS('TMatrix3x3', 'array[1..3] of TMatrix3x31');
18112: CL.AddTypeS('TMatrix4x41', 'array[1..4] of Double');
18113: CL.AddTypeS('TMatrix4x4', 'array[1..4] of TMatrix4x41');
18114: Procedure ColorTransform( A, B, C : Byte; out X, Y, Z : Byte; const T : TMatrix4x4 );
18115: Procedure ColorTransform1( A, B, C : Byte; out X, Y, Z : Float; const T : TMatrix4x4 );
18116: Procedure ColorTransform2( const A, B, C : Float; out X, Y, Z : Byte; const T : TMatrix4x4 );
18117: Procedure ColorTransformHSI2RGB( H, S, I : Byte; out R, G, B : Byte );
18118: Procedure ColorTransformRGB2HSI( R, G, B : Byte; out H, S, I : Byte );
18119: Procedure ColorTransformRGB2Lab( R, G, B : Byte; out L, a_, b_ : Byte );
18120: Procedure ColorTransformLab2RGB( L, a_, b_ : Byte; out R, G, B : Byte );
18121: Procedure ColorTransformRGB2LOCO( R, G, B : Byte; out S0, S1, S2 : Byte );
18122: Procedure ColorTransformLOCO2RGB( S0, S1, S2 : Byte; out R, G, B : Byte );
18123: Procedure ConvertColorSpace( Image : TLinarBitmap; const T : TMatrix4x4; NewImage : TLinarBitmap );
18124: //Procedure
18125: ConvertColorSpace1(Image:TLinarBitmap;ColorTransform:TColorTransformProc;NewImage:TLinarBitmap);
18126: Procedure ConvertToGrayscale( const Image, GrayImage : TLinarBitmap );
18127: Procedure ConvertToGrayscale1( const Image : TLinarBitmap );
18128:
18129: procedure SIRegister_ZDbcUtils(CL: TPSPascalCompiler);
18130: begin
18131: { TZSQLType = (zsqlstUnknown, zsqlstBoolean, zsqlstByte, zsqlstShort, zsqlstInteger, zsqlstLong,
18132: zsqlstFloat, zsqlstDouble, zsqlstBigDecimal, zsqlstString, zsqlstUnicodeString, zsqlstBytes,
18133: zsqlstDate, zsqlstTime, zsqlstTimestamp, zsqlstDataSet, zsqlstGUID,
18134: stAsciiStream, stUnicodeStream, stBinaryStream); }
18135: Function ResolveConnectionProtocol( Url : string; SupportedProtocols : TStringDynArray ) : string';
18136: //Procedure ResolveDatabaseUrl( const Url: string; Info:TStrings; var HostName:string; var
18137: Port:Integer;var Database : string; var UserName : string; var Password : string; ResultInfo : TStrings );
18138: Function CheckConversion( InitialType : TZSQLType; ResultType : TZSQLType ) : Boolean';
18139: Function DefineColumnType( ColumnType : TZSQLType ) : string';

```

```

18139: Procedure RaiseSQLEception( E : Exception)'');
18140: //Procedure CopyColumnInfo( FromList : TObjectList; ToList : TObjectList)');
18141: Function ToLikeString( const Value : string) : string');
18142: Function GetSQLHexWideString( Value : PChar; Len : Integer; ODBC : Boolean) : WideString');
18143: Function GetSQLHexAnsiString( Value : PChar; Len : Integer; ODBC : Boolean) : RawByteString');
18144: Function GetSQLHexString( Value : PChar; Len : Integer; ODBC : Boolean) : String');
18145: Function WideStringStream( const AString : WideString) : TStream');
18146: function ConvertAdoToTypeName(FieldType: SmallInt): string;
18147: function GetTableName(const AField: TField): string;');
18148: function GetFieldName(const AField: TField): string;');
18149: end;
18150:
18151: procedure SIRegister_JclTD32(CL: TPSPascalCompiler);
18152: CL.AddTypeS('TSymbolInfo', 'record Size : Word; SymbolType : Word; end');
18153: //CL.AddTypeS('PSymbolInfos', '^TSymbolInfos // will not work');
18154: SIRegister_TUclModuleInfo(CL);
18155: SIRegister_TJclLineInfo(CL);
18156: SIRegister_TUclSourceModuleInfo(CL);
18157: SIRegister_TJclSymbolInfo(CL);
18158: SIRegister_TJclProcSymbolInfo(CL);
18159: CL.AddClassN(CL.FindClass('TOBJECT'), 'TJclLocalProcSymbolInfo');
18160: CL.AddClassN(CL.FindClass('TOBJECT'), 'TJclGlobalProcSymbolInfo');
18161: SIRegister_TJclTD32InfoParser(CL);
18162: SIRegister_TJclTD32InfoScanner(CL);
18163: SIRegister_TJclPeBorTD32Image(CL);
18164: end;
18165:
18166: procedure SIRegister_JvIni(CL: TPSPascalCompiler);
18167: begin
18168: CL.AddTypeS('TReadObjectEvent', 'Function ( Sender : TObject; const Section,
18169: +Item, Value : string) : TObject');
18170: TWriteObjectEvent', 'Procedure ( Sender : TObject; const Section, Item: string; Obj: TObject)');
18171: Function StringToFontStyles( const Styles : string) : TFontStyles');
18172: Function FontStylesToString( Styles : TFontStyles) : string');
18173: Function FontToString( Font : TFont) : string');
18174: Procedure StringToFont( const Str : string; Font : TFont)');
18175: Function RectToStr( Rect : TRect) : string');
18176: Function StrToRect( const Str : string; const Def : TRect) : TRect');
18177: Function JPointToStr( P : TPoint) : string');
18178: Function JStrToPoint( const Str : string; const Def : TPoint) : TPoint');
18179: Function DefProfileName : string');
18180: Function DefLocalProfileName : string');
18181: CL.AddConstantN('idnListItem','String').SetString( 'Item');
18182: end;
18183:
18184: procedure SIRegister_JvHtControls(CL: TPSPascalCompiler);
18185: begin
18186: Procedure ItemHtDrawEx( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text :
18187: string; const HideSelColor : Boolean; var PlainItem : string; var Width : Integer; CalcWidth : Boolean)');
18188: Function ItemHtDraw( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text : string;
18189: const HideSelColor : Boolean) : string');
18190: Function ItemHtWidth( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text : string;
18191: const HideSelColor : Boolean) : Integer');
18192: Function ItemHtPlain( const Text : string) : string');
18193: Procedure ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:string);
18194: Function IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:string;MouseX,MouseY:Integer;var
18195: HyperLink:string):Bool;
18196: end;
18197: procedure SIRegister_NeuralNetwork(CL: TPSPascalCompiler);
18198: begin
18199: CL.AddClassN(CL.FindClass('TOBJECT'), 'TNeuron');
18200: CL.AddTypeS('TSynapse', 'record W : Real; Connection : TNeuron; end');
18201: CL.AddTypeS('TAcson', 'record Alfa : Real; Beta : Real; Gama : Real; end');
18202: SIRegister_TNeuron(CL);
18203: SIRegister_TNeuronLayer(CL);
18204: SIRegister_TNeuralNet(CL);
18205: end;
18206: procedure SIRegister_StExpr(CL: TPSPascalCompiler);
18207: begin
18208: //CL.AddTypeS('PStFloat', '^TStFloat // will not work');
18209: TStMethod0Param', 'Function : TStFloat');
18210: TStMethod1Param', 'Function ( Value1 : TStFloat) : TStFloat');
18211: TStMethod2Param', 'Function ( Value1, Value2 : TStFloat) : TStFloat');
18212: TStMethod3Param', 'Function ( Value1, Value2, Value3 : TStFloat) : TStFloat');
18213: TStGetIdentValueEvent', 'Procedure ( Sender : TObject; const Ide'
18214: +'ntifier : AnsiString; var Value : TStFloat)');
18215: TStToken', '( ssStart, ssInIdent, ssInNum, ssInSign, ssInExp, ss
18216: +'Eol, ssNum, ssIdent, ssLPar, ssRPar, ssComma, ssPlus, ssMinus, ssTimes, ssDiv, ssEqual, ssPower )');
18217: SIRegister_TStExpression(CL);
18218: TStExprErrorEvent', 'Procedure ( Sender : TObject; ErrorNumber : '
18219: +' LongInt; const ErrorStr : AnsiString)');
18220: Function AnalyzeExpr( const Expr : AnsiString) : Double';
18221: Procedure TpVal( const S : AnsiString; var V : Extended; var Code : Integer)');
18222: end;
18223: procedure SIRegister_GR32_Containers(CL: TPSPascalCompiler);

```

```

18224: begin
18225:   CL.AddConstantN('BUCKET_MASK','LongWord').SetUInt( $FF);
18226:   CL.AddConstantN('BUCKET_COUNT','LongInt').SetInt( BUCKET_MASK + 1);
18227:   Procedure SmartAssign( Src, Dst : TPersistent; TypeKinds : TTypeKinds );
18228:   Procedure Advance( var Node : TLinkedNode; Steps : Integer );
18229: end;
18230:
18231: procedure SIRegister_StSaturn(CL: TPSPascalCompiler);
18232: begin
18233:   TStJupSatPos', 'record X: double; Y: Double; end');
18234:   TStJupSats', 'record Io: TStJupSatPos;
18235:     Europa:TStJupSatPos;Ganymede:TStJupSatPos;Callisto:TStJupSatPos;end';
18236:   Function ComputeSaturn( JD : Double ) : TStEclipticalCord';
18237:   Function ComputePluto( JD : Double ) : TStEclipticalCord';
18238:   Function ComputeVenus( JD : Double ) : TStEclipticalCord';
18239:   Function ComputeMars( JD : Double ) : TStEclipticalCord';
18240:   Function ComputeMercury( JD : Double ) : TStEclipticalCord';
18241:   Function ComputeJupiter( JD : Double ) : TStEclipticalCord';
18242:   Function ComputeUranus( JD : Double ) : TStEclipticalCord';
18243:   Function ComputeNeptune( JD : Double ) : TStEclipticalCord';
18244:   function GetJupSats(JD : TDateTime; HighPrecision, Shadows : Boolean) : TStJupSats;');
18245:
18246: procedure SIRegister_JclParseUses(CL: TPSPascalCompiler);
18247: begin
18248:   CL.AddClassN(CL.FindClass('TOBJECT'),'EUsesListError');
18249:   Function CreateGoal( Text : PChar ) : TCustomGoal;
18250: end;
18251:
18252: procedure SIRegister_JvFinalize(CL: TPSPascalCompiler);
18253: begin
18254: //type
18255: //  TFinalizeProc = procedure;
18256:   CL.AddTypeS('TFinalizeProc', 'procedure');
18257:   Procedure AddFinalizeProc( const UnitName : string; FinalizeProc : TFinalizeProc );
18258:   Function AddFinalizeObject( const UnitName : string; Instance : TObject ) : TObject;
18259:   Function AddFinalizeObjectNil( const UnitName : string; var Reference: TObject ) : TObject;
18260:   Function AddFinalizeFreeAndNil( const UnitName : string; var Reference: TObject ) : TObject;
18261:   Function AddFinalizeMemory( const UnitName : string; Ptr : __Pointer ) : __Pointer;
18262:   Function AddFinalizeMemoryNil( const UnitName : string; var Ptr: __Pointer ) : __Pointer;
18263:   Procedure FinalizeUnit( const UnitName : string );
18264: end;
18265:
18266: procedure SIRegister_BigIni(CL: TPSPascalCompiler);
18267: begin
18268:   CL.AddConstantN('InitTextBufferSize','LongWord').SetUInt( $7000);
18269:   CL.AddConstantN('ciniCount','String').SetString( 'Count' );
18270:   TEraseSectionCallback', 'Function ( const sectionName : string; '
18271:   const sl1, sl2 : TStringList ) : Boolean';
18272:   SIRegister_TCommaSeparatedInfo(CL);
18273:   SIRegister_TSectionList(CL);
18274:   SIRegister_TBigIniFile(CL);
18275:   SIRegister_TBiggerIniFile(CL);
18276:   SIRegister_TAppIniFile(CL);
18277:   SIRegister_TLibIniFile(CL);
18278:   Function ModuleName( getLibraryName : Boolean ) : String';
18279: end;
18280:
18281: procedure SIRegister_ShellCtrls(CL: TPSPascalCompiler);
18282: begin
18283:   CL.AddTypeS('TRoot', 'string');
18284:   CL.AddTypeS('TRootFolder', '( rfDesktop, rfMyComputer, rfNetwork, rfRecycleBin
18285:   + 'n, rfAppData, rfCommonDesktopDirectory, rfCommonPrograms, rfCommonStartMen
18286:   + 'u, rfCommonStartup, rfControlPanel, rfDesktopDirectory, rfFavorites, rfFon
18287:   + 'ts, rfInternet, rfPersonal, rfPrinters, rfPrintHood, rfPrograms, rfRecent,
18288:   + ' rfSendTo, rfStartMenu, rfStartup, rfTemplates )');
18289:   TShellFolderCapability', '( fcCanCopy, fcCanDelete, fcCanLink, f
18290:   + 'cCanMove, fcCanRename, fcDropTarget, fcHasPropSheet )';
18291:   TShellFolderCapabilities', 'set of TShellFolderCapability';
18292:   TShellFolderProperty', '( fpCut, fpIsLink, fpReadOnly, fpShared,
18293:   + ' fpFileSystem, fpFileSystemAncestor, fpRemovable, fpValidate )';
18294:   TShellFolderProperties', 'set of TShellFolderProperty';
18295:   TShellObjectType', '( otFolders, otNonFolders, otHidden )';
18296:   TShellObjectTypes', 'set of TShellObjectType';
18297:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInvalidPath');
18298:   SIRegister_IShellCommandVerb(CL);
18299:   SIRegister_TShellFolder(CL);
18300:   TNotifyFilter', '( nfFileNameChange, nfDirNameChange, nfAttribut
18301:   + 'eChange, nfSizeChange, nfWriteChange, nfSecurityChange )';
18302:   TNotifyFilters', 'set of TNotifyFilter';
18303:   SIRegister_TShellChangeThread(CL);
18304:   SIRegister_TCustomShellChangeNotifier(CL);
18305:   SIRegister_TShellChangeNotifier(CL);
18306:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomShellComboBox');
18307:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomShellListView');
18308:   TAddFolderEvent', 'Procedure ( Sender : TObject; AFolder : TShel
18309:   + 'lFolder; var CanAdd : Boolean )';
18310:   TGetImageIndexEvent', 'Procedure ( Sender : TObject; Index : Int
18311:   + 'eger; var ImageIndex : Integer )';

```

```

18312:  SIRegister_TCustomShellTreeView(CL);
18313:  SIRegister_TShellTreeView(CL);
18314:  SIRegister_TCustomShellComboBox(CL);
18315:  SIRegister_TShellComboBox(CL);
18316:  SIRegister_TCustomShellListView(CL);
18317:  SIRegister_TShellListView(CL);
18318:  Procedure InvokeContextMenu( Owner : TWinControl; AFolder : TShellFolder; X, Y : Integer)'');
18319:  end;
18320:
18321: procedure SIRegister_fmath(CL: TPSPascalCompiler);
18322: begin
18323:   CL.AddTypeS('Float', 'Double');
18324:   'FN_OK','LongInt').SetInt( 0 );
18325:   CL.AddConstantN('FN_DOMAIN','LongInt').SetInt( - 1 );
18326:   'FN_SING','LongInt').SetInt( - 2 );
18327:   'FN_OVERFLOW','LongInt').SetInt( - 3 );
18328:   'FN_UNDERFLOW','LongInt').SetInt( - 4 );
18329:   'FN_TLOSS','LongInt').SetInt( - 5 );
18330:   'FN_PLOSS','LongInt').SetInt( - 6 );
18331: //CL.AddConstantN('NFACT','LongInt').SetInt( 33 );
18332:   Function MathError : Integer';
18333:   Function FMin( X, Y : Float ) : Float';
18334:   Function FMax( X, Y : Float ) : Float';
18335:   Function IMin( X, Y : Integer ) : Integer';
18336:   Function IMax( X, Y : Integer ) : Integer';
18337:   Function FSgn( X : Float ) : Integer';
18338:   Function Sgn0( X : Float ) : Integer';
18339:   Function DSgn( A, B : Float ) : Float';
18340:   Procedure FSwap( var X, Y : Float );
18341:   Procedure ISwap( var X, Y : Integer );
18342:   Function fExpo( X : Float ) : Float';
18343:   Function fExp2( X : Float ) : Float';
18344:   Function fExp10( X : Float ) : Float';
18345:   Function fLog( X : Float ) : Float';
18346:   Function fLog2( X : Float ) : Float';
18347:   Function fLog10( X : Float ) : Float';
18348:   Function fLogA( X, A : Float ) : Float';
18349:   Function fIntPower( X : Float; N : Integer ) : Float';
18350:   Function fPower( X, Y : Float ) : Float';
18351:   Function Pythag( X, Y : Float ) : Float';
18352:   Function FixAngle( Theta : Float ) : Float';
18353:   Function fTan( X : Float ) : Float';
18354:   Function fArcSin( X : Float ) : Float';
18355:   Function fArcCos( X : Float ) : Float';
18356:   Function fArcTan2( Y, X : Float ) : Float';
18357:   Procedure fSinCos( X : Float; var SinX, CosX : Float );
18358:   Function fSinh( X : Float ) : Float';
18359:   Function fCosh( X : Float ) : Float';
18360:   Function fTanh( X : Float ) : Float';
18361:   Function fArcSinh( X : Float ) : Float';
18362:   Function fArcCosh( X : Float ) : Float';
18363:   Function fArcTanh( X : Float ) : Float';
18364:   Procedure fSinhCosh( X : Float; var SinhX, CoshX : Float );
18365:   Function fFact( N : Integer ) : Float';
18366:   Function fBinomial( N, K : Integer ) : Float';
18367:   Function fGamma( X : Float ) : Float';
18368:   Function fSgnGamma( X : Float ) : Integer';
18369:   Function LnGamma( X : Float ) : Float';
18370:   Function fIGamma( A, X : Float ) : Float';
18371:   Function fJGamma( A, X : Float ) : Float';
18372:   Function fBeta( X, Y : Float ) : Float';
18373:   Function fIBeta( A, B, X : Float ) : Float';
18374:   Function fErf( X : Float ) : Float';
18375:   Function fErfc( X : Float ) : Float';
18376:   Function fPBinom(N: Integer; P: Float; K : Integer) : Float';
18377:   Function FBinom(N: Integer; P: Float; K : Integer) : Float';
18378:   Function PPoisson( Mu : Float; K : Integer ) : Float';
18379:   Function FPoisson( Mu : Float; K : Integer ) : Float';
18380:   Function fDNorm( X : Float ) : Float';
18381:   Function FNorm( X : Float ) : Float';
18382:   Function PNorm( X : Float ) : Float';
18383:   Function InvNorm( P : Float ) : Float';
18384:   Function fDStudent( Nu : Integer; X : Float ) : Float';
18385:   Function FStudent( Nu : Integer; X : Float ) : Float';
18386:   Function PStudent( Nu : Integer; X : Float ) : Float';
18387:   Function fDKhi2( Nu : Integer; X : Float ) : Float';
18388:   Function FKhi2( Nu : Integer; X : Float ) : Float';
18389:   Function PKhi2( Nu : Integer; X : Float ) : Float';
18390:   Function fDSnedecor( Nul, Nu2 : Integer; X : Float ) : Float';
18391:   Function FSnedecor( Nul, Nu2 : Integer; X : Float ) : Float';
18392:   Function PSnedecor( Nul, Nu2 : Integer; X : Float ) : Float';
18393:   Function fDExpo( A, X : Float ) : Float';
18394:   Function FExpo( A, X : Float ) : Float';
18395:   Function fDBeta( A, B, X : Float ) : Float';
18396:   Function FBeta( A, B, X : Float ) : Float';
18397:   Function fDGamma( A, B, X : Float ) : Float';
18398:   Function FGamma( A, B, X : Float ) : Float';
18399:   Procedure RMarIn( Seed1, Seed2 : Integer );
18400:   Function IRanMar : LongInt );

```

```

18401: Function RanMar : Float');
18402: Function RanGaussStd : Float');
18403: Function RanGauss( Mu, Sigma : Float) : Float');
18404: end;
18405:
18406: procedure SIRegister_fcomp(CL: TPSPascalCompiler);
18407: begin
18408: CL.AddTypeS('ComplexForm', '( Rec, Pol )');
18409: CL.AddTypeS('TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R :'
18410: +' Float; Theta : Float; end');
18411: Procedure CSet( var Z : TComplex; A, B : Float; F : ComplexForm)');
18412: Procedure CConvert( var Z : TComplex; F : ComplexForm)');
18413: Procedure CSwap( var X, Y : TComplex)');
18414: Function CReal( Z : TComplex) : Float');
18415: Function CIImag( Z : TComplex) : Float');
18416: Function CAbs( Z : TComplex) : Float');
18417: Function CArg( Z : TComplex) : Float');
18418: Function CSgn( Z : TComplex) : Integer');
18419: Procedure CNeg( A : TComplex; var Z : TComplex)');
18420: Procedure CConj( A : TComplex; var Z : TComplex)');
18421: Procedure CAdd( A, B : TComplex; var Z : TComplex)');
18422: Procedure CSub( A, B : TComplex; var Z : TComplex)');
18423: Procedure CDiv( A, B : TComplex; var Z : TComplex)');
18424: Procedure CMult( A, B : TComplex; var Z : TComplex)');
18425: Procedure CLn( A : TComplex; var Z : TComplex)');
18426: Procedure CExp( A : TComplex; var Z : TComplex)');
18427: Procedure CPower( A, B : TComplex; var Z : TComplex)');
18428: Procedure CIntPower( A : TComplex; N : Integer; var Z : TComplex)');
18429: Procedure CRealPower( A : TComplex; X : Float; var Z : TComplex)');
18430: Procedure CSqrt( A : TComplex; var Z : TComplex)');
18431: Procedure CRoot( A : TComplex; K, N : Integer; var Z : TComplex)');
18432: Procedure CCSin( A : TComplex; var Z : TComplex)');
18433: Procedure CCos( A : TComplex; var Z : TComplex)');
18434: Procedure CTan( A : TComplex; var Z : TComplex)');
18435: Procedure CArcSin( A : TComplex; var Z : TComplex)');
18436: Procedure CArcCos( A : TComplex; var Z : TComplex)');
18437: Procedure CArcTan( A : TComplex; var Z : TComplex)');
18438: Procedure CSinh( A : TComplex; var Z : TComplex)');
18439: Procedure CCosh( A : TComplex; var Z : TComplex)');
18440: Procedure CTanh( A : TComplex; var Z : TComplex)');
18441: Procedure CArcSinh( A : TComplex; var Z : TComplex)');
18442: Procedure CArcCosh( A : TComplex; var Z : TComplex)');
18443: Procedure CArcTanh( A : TComplex; var Z : TComplex)');
18444: Procedure CLnGamma( A : TComplex; var Z : TComplex)');
18445: end;
18446:
18447: procedure SIRegister_XSBuiltIns(CL: TPSPascalCompiler);
18448: begin
18449: CL.AddConstantN('SHexMarker','String').SetString( '$');
18450: Function DateTimeToXMLTime( Value : TDateTime; ApplyLocalBias : Boolean) : WideString';
18451: Function XMLTimeToDate( const XMLDate : WideString; AsUTCTime : Boolean): TDateTime';
18452: Function DateTimeToXSDateTime(const Value: TDateTime; ApplyLocalBias: Boolean): TXSDateTime';
18453: Function GetDataFromFile( AFileName : string) : string');
18454: Function SoapFloatToStr( Value : double) : string');
18455: Function SoapStrToFloat( Value : string) : double');
18456: Procedure InitXSTypes');
18457: end;
18458:
18459:
18460: function TRestRequest_createStringStreamFromArrayList(strings: TStringList): TFileStream;
18461:
18462: {A simple Oscilloscope using TWaveIn class.
18463: More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
18464: http://www.retroarchive.org/garbo/pc/turbopas/index.html
18465: uses
18466:   Forms,
18467:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
18468:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
18469:   uColorFunctions in 'uColorFunctions.pas',
18470:   AMixer in 'AMixer.pas',
18471:   uSettings in 'uSettings.pas',
18472:   UWavein4 in 'UWavein4.pas',
18473:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
18474:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
18475:
18476: Functions_max hex in the box maxbox
18477: functionslist.txt
18478: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98/100/101/110/120/160/180/190
18479:
18480: ****
18481: Procedure
18482: PROCEDURE SIZE 8875 8396 8289 8242 7507 7401 6792 6310 5971 4438 3797 3600
18483: Procedure *****Now the Procedure list*****
18484: Procedure ( ACol, ARow : Integer; Items : TStrings)
18485: Procedure ( Agg : TAggregate)
18486: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
18487: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
18488: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
18489: Procedure ( ASender : TObject; const ABytes : Integer)

```

```

18490: Procedure ( ASender : TObject; VStream : TStream)
18491: Procedure ( AThread : TIdThread)
18492: Procedure ( AWebModule : TComponent)
18493: Procedure ( Column : TColumn)
18494: Procedure ( const AUsername : String; const APASSWORD : String; AAAuthenticationResult : Boolean)
18495: Procedure ( const iStart : integer; const sText : string)
18496: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
18497: Procedure ( Database : TDatabase; LoginParams : TStrings)
18498: Procedure ( DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:TReconcileAction)
18499: Procedure ( DATASET : TDATASET)
18500: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
18501: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
18502: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
18503: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
18504: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
18505: Procedure ( Done : Integer)
18506: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
18507: Procedure ( HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
18508: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
18509: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
18510: Procedure ( HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
18511: Procedure ( HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
18512: Procedure ( Sender : TCustomListView;const ARect : TRect;Stage:TCustomDrawStage;var DefaultDraw : Boolean)
18513: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
18514: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
18515: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
18516: Procedure ( SENDER : TFIELD; const TEXT : String)
18517: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : Boolean)
18518: Procedure ( Sender : TIdTelnet; const Buffer : String)
18519: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
18520: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : Boolean)
18521: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18522: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : Integer)
18523: Procedure ( Sender : TObj; ARow : Longint; const Value : string)
18524: Procedure ( Sender : TObj; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
18525: Procedure ( Sender : TObj; ACol, ARow : Longint; var CanSelect : Boolean)
18526: Procedure ( Sender : TObj; ACol, ARow : Longint; var Value : string)
18527: Procedure ( Sender : TObj; Button : TMPBnType)
18528: Procedure ( Sender : TObj; Button : TMPBnType; var DoDefault : Boolean)
18529: Procedure ( Sender : TObj; Button : TUDBnType)
18530: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
18531: Procedure ( Sender : TObject; ClientSocket : TServerClientWinSocket; var SocketThread : TServerClientThread)
18532: Procedure ( Sender : TObject; Column : TListColumn)
18533: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
18534: Procedure ( Sender : TObject; Connecting : Boolean)
18535: Procedure ( Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var DoneDraw:Bool)
18536: Procedure ( Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
18537: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
18538: Procedure ( Sender: TObject; const UserString:string;var DateAndTime:TDateAndTime;var AllowChange:Boolean)
18539: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
18540: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
18541: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
18542: Procedure ( Sender : TObject; Index : LongInt)
18543: Procedure ( Sender : TObject; Item : TListItem)
18544: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
18545: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
18546: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
18547: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
18548: Procedure ( Sender : TObject; Item : TListItem; var S : string)
18549: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
18550: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
18551: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
18552: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
18553: Procedure ( Sender : TObject; Node : TTreenode)
18554: Procedure ( Sender : TObject; Node : TTreenode; var AllowChange : Boolean)
18555: Procedure ( Sender : TObject; Node : TTreenode; var AllowCollapse : Boolean)
18556: Procedure ( Sender : TObject; Node : TTreenode; var AllowEdit : Boolean)
18557: Procedure ( Sender : TObject; Node : TTreenode; var AllowExpansion : Boolean)
18558: Procedure ( Sender : TObject; Node : TTreenode; var S : string)
18559: Procedure ( Sender : TObject; Node1, Node2 : TTreenode; Data : Integer; var Compare : Integer)
18560: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
18561: Procedure ( Sender : TObject; Rect : TRect)
18562: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
18563: Procedure ( Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int)
18564: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
18565: Procedure ( Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
18566: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
18567: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
18568: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : Boolean)
18569: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
18570: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
18571: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
18572: Procedure ( Sender : TObject; Thread : TServerClientThread)
18573: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
18574: Procedure ( Sender : TObject; Username, Password : string)
18575: Procedure ( Sender : TObject; var AllowChange : Boolean)
18576: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)

```

```

18577: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment )
18578: Procedure ( Sender : TObject; var Continue : Boolean )
18579: Procedure ( Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMETHOD:TIdHTTPMethod )
18580: Procedure ( Sender : TObject; var Username : string )
18581: Procedure ( Sender : TObject; Wnd : HWND )
18582: Procedure ( Sender : TToolbar; Button : TToolButton )
18583: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean )
18584: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean )
18585: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean )
18586: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton )
18587: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect )
18588: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect )
18589: Procedure ( var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString )
18590: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString )
18591: procedure (Sender: TObject)
18592: procedure (Sender: TObject; var Done: Boolean)
18593: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
18594: procedure _T(Name: tbtString; v: Variant);
18595: Procedure AbandonSignalHandler( RtlSigNum : Integer )
18596: Procedure Abort
18597: Procedure About1Click( Sender : TObject )
18598: Procedure Accept( Socket : TSocket )
18599: Procedure AESSymmetricExecute(const plaintext, ciphertext, password: string)
18600: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
18601: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
18602: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
18603: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
18604: Procedure Add(Addend1, Addend2 : TMBigInt)
18605: Procedure ADD( const AKEY, AVALUE : VARIANT )
18606: Procedure Add( const Key : string; Value : Integer )
18607: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS )
18608: Procedure ADD( FIELD : TFIELD )
18609: Procedure ADD( ITEM : TMENUITEM )
18610: Procedure ADD( POPUP : TPOPUPMENU )
18611: Procedure AddCharacters( xCharacters : TCharSet )
18612: Procedure AddDriver( const Name : string; List : TStrings )
18613: Procedure AddImages( Value : TCustomImageList )
18614: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string )
18615: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState )
18616: Procedure AddLoader( Loader : TBitmapLoader )
18617: Procedure ADDPARAM( VALUE : TPARAM )
18618: Procedure AddPassword( const Password : string )
18619: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string )
18620: Procedure AddState( oState : TniRegularExpressionState )
18621: Procedure AddStrings( Strings : TStrings );
18622: procedure AddStrings(Strings: TStrings);
18623: Procedure AddStrings1( Strings : TWideStrings );
18624: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string )
18625: Procedure AddToRecentDocs( const Filename : string )
18626: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset )
18627: Procedure AllFunctionsList1Click( Sender : TObject )
18628: procedure AllObjectsList1Click(Sender: TObject);
18629: Procedure Allocate( AAllocateBytes : Integer )
18630: procedure AllResourceList1Click(Sender: TObject);
18631: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString )
18632: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString )
18633: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer )
18634: Procedure AnsiFree( var s : AnsiString )
18635: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString )
18636: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer )
18637: Procedure AnsiSetLength( var dst : AnsiString; len : Integer )
18638: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream )
18639: Procedure AntiFreeze;
18640: Procedure APPEND
18641: Procedure Append( const S : WideString )
18642: procedure Append(S: string);
18643: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte )
18644: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes )
18645: Procedure AppendChunk( Val : OleVariant )
18646: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean )
18647: Procedure AppendStr( var Dest : string; S : string )
18648: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALen : Integer )
18649: Procedure ApplyRange
18650: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18651: Procedure Arrange( Code : TListArrangement )
18652: procedure Assert(expr : Boolean; const msg: string);
18653: procedure Assert2(expr : Boolean; const msg: string);
18654: Procedure Assign( Alist : TCustomBucketList )
18655: Procedure Assign( Other : TObject )
18656: Procedure Assign( Source : TDragObject )
18657: Procedure Assign( Source : TPersistent )
18658: Procedure Assign(Source: TPersistent)
18659: procedure Assign2(mystring, mypath: string);
18660: Procedure AssignCurValues( Source : TDataSet );
18661: Procedure AssignCurValues1( const CurValues : Variant );
18662: Procedure ASSIGNFIELD( FIELD : TFIELD )
18663: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT )
18664: Procedure AssignFile(var F: Text; FileName: string)
18665: procedure AssignFile(var F: TextFile; FileName: string)

```

```

18666: procedure AssignFileRead(var mystring, myfilename: string);
18667: procedure AssignFileWrite(mystring, myfilename: string);
18668: Procedure AssignTo( Other : TObject)
18669: Procedure AssignValues( Value : TParameters)
18670: Procedure ASSIGNVALUES( VALUE : TPARAMS)
18671: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
18672: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
18673: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
18674: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
18675: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18676: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
18677: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
18678: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
18679: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
18680: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
18681: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
18682: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
18683: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
18684: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18685: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
18686: procedure Beep
18687: Procedure BeepOk
18688: Procedure BeepQuestion
18689: Procedure BeepHand
18690: Procedure BeepExclamation
18691: Procedure BeepAsterisk
18692: Procedure BeepInformation
18693: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
18694: Procedure BeginLayout
18695: Procedure BeginTimer( const Delay, Resolution : Cardinal)
18696: Procedure BeginUpdate
18697: procedure BeginUpdate;
18698: procedure BigScreen1Click(Sender: TObject);
18699: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
18700: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
18701: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
18702: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
18703: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
18704: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
18705: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
18706: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
18707: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
18708: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
18709: Procedure BreakPointMenuClick( Sender : TObject)
18710: procedure BRINGTOFRONT
18711: procedure BringToFront;
18712: Procedure btnBackClick( Sender : TObject)
18713: Procedure btnBrowseClick( Sender : TObject)
18714: Procedure BtnClick( Index : TNavigateBtn)
18715: Procedure btnLargeIconsClick( Sender : TObject)
18716: Procedure BuildAndSendRequest( AURI : TIdURI)
18717: Procedure BuildCache
18718: Procedure BurnMemory( var Buff, BuffLen : integer)
18719: Procedure BurnMemoryStream( Destructo : TMemoryStream)
18720: Procedure CalculateFirstSet
18721: Procedure Cancel
18722: procedure CancelDrag;
18723: Procedure CancelEdit
18724: procedure CANCELHINT
18725: Procedure CancelRange
18726: Procedure CancelUpdates
18727: Procedure CancelWriteBuffer
18728: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool);
18729: Procedure Capture2(ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
18730: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool)
18731: procedure CaptureScreenFormat(vname: string; vextension: string);
18732: procedure CaptureScreenPNG(vname: string);
18733: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
18734: procedure CASCADE
18735: Procedure CastNativeToSoap(Info:PTTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
18736: Procedure CastSoapToVariant( SoapInfo : PTTypeInfo; const SoapData : WideString; NatData : Pointer);
18737: Procedure cbPathClick( Sender : TObject)
18738: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18739: Procedure cedbugAfterExecute( Sender : TPSScript)
18740: Procedure cedbugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18741: Procedure cedbugCompile( Sender : TPSScript)
18742: Procedure cedbugExecute( Sender : TPSScript)
18743: Procedure cedbugIdle( Sender : TObject)
18744: Procedure cedbugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18745: Procedure CenterHeight( const pc, pcParent : TControl)
18746: Procedure CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
18747: Procedure CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
18748: Procedure Change
18749: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
18750: Procedure Changed
18751: Procedure ChangeDir( const ADirName : string)
18752: Procedure ChangeDirUp
18753: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
18754: Procedure ChangeLevelBy( Value : TChangeRange)

```

```

18755: Procedure ChDir( const s: string )
18756: Procedure Check(Status: Integer)
18757: Procedure CheckCommonControl( CC : Integer )
18758: Procedure CHECKFIELDNAME( const FIELDNAME : String )
18759: Procedure CHECKFIELDNAMES( const FIELDNAMES : String )
18760: Procedure CheckForDisconnect( const ARaiseExceptionIfDisconnected: boolean; const AIgnoreBuffer: bool )
18761: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean )
18762: Procedure CheckToken( T : Char )
18763: procedure CheckToken(t:char)
18764: Procedure CheckTokenSymbol( const S : string )
18765: procedure CheckTokenSymbol(s:string)
18766: Procedure CheckToolMenuDropdown( ToolButton : TToolButton )
18767: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer );
18768: Procedure CIED65ToCIED50( var X, Y, Z : Extended )
18769: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal );
18770: procedure CipherFileClick(Sender: TObject);
18771: Procedure Clear;
18772: Procedure Clear1Click( Sender : TObject )
18773: Procedure ClearColor( Color : TColor )
18774: Procedure CLEARITEM( AITEM : TMENUITEM )
18775: Procedure ClearMapping
18776: Procedure ClearSelection( KeepPrimary : Boolean )
18777: Procedure ClearWriteBuffer
18778: Procedure Click
18779: Procedure Close
18780: Procedure Close1Click( Sender : TObject )
18781: Procedure CloseDatabase( Database : TDatabase )
18782: Procedure CloseDataSets
18783: Procedure CloseDialog
18784: Procedure CloseFile(var F: Text );
18785: Procedure Closure
18786: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal );
18787: Procedure CMYKTobGRI( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal );
18788: Procedure CodeCompletionList1Click( Sender : TObject )
18789: Procedure ColEnter
18790: Procedure Collapse
18791: Procedure Collapse( Recurse : Boolean )
18792: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word )
18793: Procedure CommaSeparatedToStringList( Alist : TStringList; const Value : string )
18794: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction )
18795: Procedure Compile1Click( Sender : TObject )
18796: procedure ComponentCount1Click(Sender: TObject);
18797: Procedure Compress(azipfolder, azipfile: string)
18798: Procedure DeCompress(azipfolder, azipfile: string)
18799: Procedure XZip(azipfolder, azipfile: string)
18800: Procedure XUnZip(azipfolder, azipfile: string)
18801: Procedure Connect( const ATimeout : Integer )
18802: Procedure Connect( Socket : TSocket )
18803: procedure ConsoleClick(Sender: TObject );
18804: Procedure Continue
18805: Procedure ContinueCount( var Counter : TJclCounter )
18806: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
18807: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer )
18808: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer )
18809: Procedure ConvertImage(vsource, vdestination: string);
18810: // Ex. ConvertImage(Exepath+'my233.bmp',Exepath+'mypng111.png')
18811: Procedure ConvertBitmap(vsource, vdestination: string);
18812: Procedure ConvertToGray(Cnv: TCanvas);
18813: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer )
18814: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer );
18815: Procedure Copyl( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer );
18816: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASrcIndex:Integer;var VDest:LongWord)
18817: Procedure CopyBytesToHostWord( const ASrc : TIdBytes; const ASrcIndex : Integer; var VDest : Word )
18818: Procedure CopyFrom( mbCopy : TMyBigInt )
18819: Procedure CopyMemoryStream( Source, Destination : TMemoryStream )
18820: procedure CopyRect( const Dest: TRect; Canvas: TCanvas; const Source: TRect );
18821: Procedure CopyTidByteArray( const ASrc : array of Byte; const ASrcIndex : Integer; var VDest : array of Byte; const ADestIndex : Integer; const ALen : Integer )
18822: Procedure CopyTidBytes( const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const ALen:Int )
18823: Procedure CopyTidCardinal( const ASrc : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer )
18824: Procedure CopyTidInt64( const ASrc : Int64; var VDest : TIdBytes; const ADestIndex : Integer )
18825: Procedure CopyTidIPv6Address(const ASrc:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer )
18826: Procedure CopyTidLongWord( const ASrc : LongWord; var VDest : TIdBytes; const ADestIndex : Integer )
18827: Procedure CopyTidNetworkLongWord( const ASrc : LongWord; var VDest : TIdBytes; const ADestIndex:Integer )
18828: Procedure CopyTidNetworkWord( const ASrc : Word; var VDest : TIdBytes; const ADestIndex : Integer )
18829: Procedure CopyTidString(const ASrc:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer )
18830: Procedure CopyTidWord( const ASrc : Word; var VDest : TIdBytes; const ADestIndex : Integer )
18831: Procedure CopyToClipboard
18832: Procedure CountParts
18833: Procedure CreateDataSet
18834: Procedure CreateEmptyFile( const FileName : string )
18835: Procedure CreateFileFromString( const FileName, Data : string )
18836: Procedure CreateFromDelta( Source : TPacketDataSet )
18837: procedure CREATEHANDLE
18838: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
18839: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string )
18840: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar )

```

```

18841: Procedure CreateTable
18842: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
18843: procedure CSyntax1Click(Sender: TObject);
18844: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
18845: Procedure CURSORPOSCHANGED
18846: procedure CutFirstDirectory(var S: String)
18847: Procedure DataBaseError(const Message: string)
18848: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
18849: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
18850: Procedure DateTimeToSystemTime( DateTime: TDateTime; var SystemTime : TSystemTime)
18851: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
18852: Procedure DBIError(errorCode: Integer)
18853: Procedure DebugOutput( const AText : string)
18854: procedure DebugIn(DebugLOGFILE: string; Event_message: string);
18855: Procedure DebugRun1Click( Sender : TObject)
18856: procedure Dec;
18857: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
18858: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
18859: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
18860: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
18861: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,Aday,AMin,ASec,AMillSec:Word)
18862: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
18863: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
18864: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
18865: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
18866: Procedure Decompile1Click( Sender : TObject)
18867: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
18868: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
18869: Procedure DeferLayout
18870: Procedure defFileRead
18871: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL; REMOVING:BOOLEAN)
18872: Procedure DelayMicroseconds( const MicroSeconds : Integer)
18873: Procedure Delete
18874: Procedure Delete( const Afilename : string)
18875: Procedure Delete( const Index : Integer)
18876: Procedure DELETE( INDEX : INTEGER)
18877: Procedure Delete( Index : LongInt)
18878: Procedure Delete( Node : TTreeNode)
18879: procedure Delete(var s: AnyString; ifrom, icount: Longint);
18880: Procedure DeleteAlias( const Name : string)
18881: Procedure DeleteDriver( const Name : string)
18882: Procedure DeleteIndex( const Name : string)
18883: Procedure DeleteKey( const Section, Ident : String)
18884: Procedure DeleteRecords
18885: Procedure DeleteRecords( AffectRecords : TAffectRecords)
18886: Procedure DeleteString( var pStr : String; const pDelStr : string)
18887: Procedure DeleteTable
18888: procedure DelphiSite1Click(Sender: TObject);
18889: Procedure Deselect
18890: Procedure Deselect( Node : TTreeNode)
18891: procedure DestroyComponents
18892: Procedure DestroyHandle
18893: Procedure Diff( var X : array of Double)
18894: procedure Diff(var X: array of Double);
18895: Procedure DirCreate( const DirectoryName : String)!;
18896: procedure DISABLEALIGN
18897: Procedure DisableConstraints
18898: Procedure Disconnect
18899: Procedure Disconnect( Socket : TSocket)
18900: Procedure Dispose
18901: procedure Dispose(P: PChar)
18902: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
18903: Procedure DoKey( Key : TDBCtrlGridKey)
18904: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18905: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18906: Procedure Dormant
18907: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
18908: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
18909: Procedure DoubleToComp( Value : Double; var Result : Comp)
18910: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
18911: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
18912: procedure Draw(X, Y: Integer; Graphic: TGraphic);
18913: Procedure Draw1(Canvas:TCanvas; X,Y,
Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
18914: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18915: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
18916: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18917: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
18918: procedure DrawFocusRect(const Rect: TRect);
18919: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
18920: Procedure DRAWMENUEITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18921: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
18922: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle : TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
18923: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
18924: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
18925: Procedure DropConnections
18926: Procedure DropDown
18927: Procedure DumpDescription( oStrings : TStrings)

```

```

18928: Procedure DumpStateTable( oStrings : TStrings )
18929: Procedure EDIT
18930: Procedure EditButtonClick
18931: Procedure EditFontClick( Sender : TObject )
18932: procedure Ellipse(X1, Y1, X2, Y2: Integer);
18933: Procedure Ellipse( const Rect : TRect );
18934: Procedure EMMS
18935: Procedure Encode( ADest : TStream )
18936: procedure ENDDRAG(DROP:BOOLEAN)
18937: Procedure EndEdit( Cancel : Boolean )
18938: Procedure EndTimer
18939: Procedure EndUpdate
18940: Procedure EraseSection( const Section : string )
18941: Procedure Error( const Ident : string )
18942: procedure Error(Ident:Integer)
18943: Procedure ErrorFmt( const Ident : string; const Args : array of const )
18944: Procedure ErrorStr( const Message : string )
18945: procedure ErrorStr(Message:String)
18946: Procedure Exchange( Index1, Index2 : Integer )
18947: procedure Exchange(Index1, Index2: Integer);
18948: Procedure Exec( FileName, Parameters, Directory : string )
18949: Procedure ExecProc
18950: Procedure ExecSQL( UpdateKind : TUpdateKind )
18951: Procedure Execute
18952: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant )
18953: Procedure ExecuteAndWait( FileName : string; Visibility : Integer )
18954: Procedure ExecuteCommand(executeFile, paramstring: string)
18955: Procedure ExecuteShell(executeFile, paramstring: string)
18956: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18957: Procedure ExitThread(ExitCode: Integer); stdcall;
18958: Procedure ExitProcess(ExitCode: Integer); stdcall;
18959: Procedure Expand( AUserName : String; AResults : TStrings )
18960: Procedure Expand( Recurse : Boolean )
18961: Procedure ExportClipboardClick( Sender : TObject )
18962: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress )
18963: Procedure ExtractContentFields( Strings : TStrings )
18964: Procedure ExtractCookieFields( Strings : TStrings )
18965: Procedure ExtractFields( Separators,WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings )
18966: Procedure ExtractHeaderFields(Separ,
WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
18967: Procedure ExtractHTTPFields(Separators,WhiteSpace:
TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
18968: Procedure ExtractQueryFields( Strings : TStrings )
18969: Procedure FastDegToGrad
18970: Procedure FastDegToRad
18971: Procedure FastGradToDeg
18972: Procedure FastGradToRad
18973: Procedure FastRadToDeg
18974: Procedure FastRadToGrad
18975: Procedure FileClose( Handle : Integer )
18976: Procedure FileClose(handle: integer)
18977: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
18978: Procedure FileStructure( AStructure : TIIdFTPDataStructure )
18979: Procedure FillByte2(var X: Byte; count: integer; value: byte)
18980: Procedure FillBytes( var VBytes : TIBytes; const ACount : Integer; value : Byte )
18981: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte )
18982: Procedure FillChar2(var X: PChar ; count: integer; value: char)
18983: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
18984: Procedure FillIPList
18985: procedure FillRect(const Rect: TRect);
18986: Procedure FillTStrings( AStrings : TStrings )
18987: Procedure FilterOnBookmarks( Bookmarks : array of const )
18988: procedure FinalizePackage(Module: HMODULE)
18989: procedure FindClose;
18990: procedure FindClose2(var F: TSearchRec)
18991: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent );
18992: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent );
18993: Procedure FindNearest( const KeyValues : array of const )
18994: Procedure FinishContext
18995: Procedure FIRST
18996: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float )
18997: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
18998: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle )
18999: Procedure FlushSchemaCache( const TableName : string )
19000: procedure FmtStr( var Result: string; const Format: string; const Args: array of const )
19001: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
19002: Procedure Form1Close( Sender : TObject; var Action : TCloseAction )
19003: Procedure FormActivate( Sender : TObject )
19004: procedure FormatLn(const format: String; const args: array of const); //alias
19005: Procedure FormClose( Sender : TObject; var Action : TCloseAction )
19006: Procedure FormCreate( Sender : TObject )
19007: Procedure FormDestroy( Sender : TObject )
19008: Procedure FormKeyPress( Sender : TObject; var Key : Char )
19009: procedure FormOutput1Click(Sender: TObject);
19010: Procedure FormToHtml( Form : TForm; Path : string )
19011: procedure FrameRect(const Rect: TRect);
19012: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
19013: Procedure NotebookHandlesNeeded( Notebook : TNotebook )
19014: Procedure Free( Buffer : TRecordBuffer )

```

```

19015: Procedure Free( Buffer : TValueBuffer )
19016: Procedure Free;
19017: Procedure FreeAndNil(var Obj:TObject)
19018: Procedure FreeImage
19019: procedure FreeMem(P: PChar; Size: Integer)
19020: Procedure FreeTreeData( Tree : TUpdateTree)
19021: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
19022: Procedure FullCollapse
19023: Procedure FullExpand
19024: Procedure GenerateDB(sl: TStrings; var DPB: string; var DPBLength: Short); //InterBase
19025: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
19026: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBPUTPUTXML)
19027: Procedure Get1( AURL : string; const AResponseContent : TStream);
19028: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
19029: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
19030: Procedure GetAliasNames( List : TStrings)
19031: Procedure GetAliasParams( const AliasName : string; List : TStrings)
19032: Procedure GetApplicationsRunning( Strings : TStrings)
19033: Procedure getBox(aURL, extension: string);
19034: Procedure GetCommandTypes( List : TWideStrings)
19035: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
19036: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
19037: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
19038: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
19039: Procedure GetDatabaseNames( List : TStrings)
19040: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
19041: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
19042: Procedure GetDir(d: byte; var s: string)
19043: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
19044: Procedure GetDriverNames( List : TStrings)
19045: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
19046: Procedure GetDriverParams( const DriverName : string; List : TStrings)
19047: Procedure GetEmailsOnClick( Sender : TObject)
19048: Procedure getEnvironmentInfo;
19049: Function getEnvironmentString: string;
19050: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
19051: Procedure GetFieldNames( const TableName : string; List : TStrings)
19052: Procedure GetFieldNames( const TableName : string; List : TStrings);
19053: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
19054: Procedure GETFIELDNAMES( LIST : TSTRINGS)
19055: Procedure GetFieldNames1( const TableName : string; List : TStrings);
19056: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
19057: Procedure GetFieldNames2( const TableName : WideString; SchemaName : WideString; List : TWideStrings);
19058: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
19059: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
19060: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
19061: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
19062: Procedure GetFormatSettings
19063: Procedure GetFromDIB( var DIB : TBitmapInfo)
19064: Procedure GetFromHDI( HDIB : HBitmap)
19065: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
19066: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
19067: Procedure GetIcon( Index : Integer; Image : TIcon);
19068: Procedure GetIconl(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
19069: Procedure GetIndexInfo( IndexName : string)
19070: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
19071: Procedure GetIndexNames( List : TStrings)
19072: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
19073: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
19074: Procedure GetIndexNames4( const TableName : string; List : TStrings);
19075: Procedure GetInternalResponse
19076: Procedure GETITEMNAMES( LIST : TSTRINGS)
19077: procedure GetMem(P: PChar; Size: Integer)
19078: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
19079: procedure GetPackageDescription(ModuleName: PChar): string)
19080: Procedure GetPackageNames( List : TStrings);
19081: Procedure GetPackageNames1( List : TWideStrings);
19082: Procedure GetParamList( List : TList; const ParamNames : WideString)
19083: Procedure GetProcedureNames( List : TStrings);
19084: Procedure GetProcedureNames( List : TWideStrings);
19085: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
19086: Procedure GetProcedureNames1( List : TStrings);
19087: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
19088: Procedure GetProcedureNames3( List : TWideStrings);
19089: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
19090: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
19091: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
19092: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : Tlist);
19093: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
19094: Procedure GetProviderNames( Names : TWideStrings);
19095: Procedure GetProviderNames( Proc : TGetStrProc)
19096: Procedure GetProviderNames1( Names : TStrings);
19097: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
19098: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string); //no open image
19099: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const Data:string;format:string):TLinearBitmap;
19100: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
19101: Procedure GetSchemaNames( List : TStrings);
19102: Procedure GetSchemaNames1( List : TWideStrings);

```

```

19103: Procedure getScriptandRunAsk;
19104: Procedure getScriptandRun(ascript: string);
19105: Procedure getScript(ascript: string); //alias
19106: Procedure getWebScript(ascript: string); //alias
19107: Procedure GetSessionNames( List : TStrings )
19108: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings )
19109: Procedure GetStrings( List : TStrings )
19110: Procedure GetSystemTime; stdcall;
19111: Procedure GetTableNames( const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings )
19112: Procedure GetTableNames( List : TStrings; SystemTables : Boolean )
19113: Procedure GetTableNames( List : TStrings; SystemTables : Boolean );
19114: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean );
19115: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean );
19116: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean );
19117: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean );
19118: Procedure GetTransitionsOn( cChar : char; oStateList : TList )
19119: Procedure GetVisibleWindows( List : TStrings )
19120: Procedure GoBegin
19121: Procedure GotoCurrent( DataSet : TCustomClientDataSet )
19122: Procedure GotoCurrent( Table : TTable )
19123: procedure GotoEnd1Click(Sender: TObject);
19124: Procedure GotoNearest
19125: Procedure GradientFillCanvas(const ACanvas: TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
Direction: TGradientDirection)
19126: Procedure HandleException( E : Exception; var Handled : Boolean )
19127: procedure HANDLEMESSAGE
19128: procedure HandleNeeded;
19129: Procedure Head( AURL : string )
19130: Procedure Help( var AHelpContents : TStringList; ACommand : String )
19131: Procedure HexToBinary( Stream : TStream )
19132: procedure HexToBinary(Stream:TStream)
19133: Procedure HideDragImage
19134: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean )
19135: Procedure HideTraybar
19136: Procedure HideWindowForSeconds(secs: integer); //3 seconds
19137: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); //3 seconds
19138: Procedure HookOSExceptions
19139: Procedure HookSignal( RtlSigNum : Integer )
19140: Procedure HSLTtoRGB( const H, S, L : Single; out R, G, B : Single );
19141: Procedure HTMLSyntax1Click( Sender : TObject )
19142: Procedure IFPS3ClassesPluginInlCompImport( Sender : TObject; x : TPSPascalCompiler )
19143: Procedure IFPS3ClassesPluginInlExecImport( Sender : TObject; Exec : TPSEexec; x : TPSRuntimeClassImporter )
19144: Procedure ImportfromClipboard1Click( Sender : TObject )
19145: Procedure ImportfromClipboard2Click( Sender : TObject )
19146: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer )
19147: procedure Incb(var x: byte);
19148: Procedure Include1Click( Sender : TObject )
19149: Procedure IncludeOFF; //preprocessing
19150: Procedure IncludeON;
19151: procedure Info1Click(Sender: TObject);
19152: Procedure InitAltRecBuffers( CheckModified : Boolean )
19153: Procedure InitContext( Request : TWebRequest; Response : TWebResponse )
19154: Procedure InitContext( WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse )
19155: Procedure InitData( ASource : TDataSet )
19156: Procedure InitDelta( ADelta : TPacketDataSet );
19157: Procedure InitDelta( const ADelta : OleVariant );
19158: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse )
19159: Procedure Initialize
19160: procedure InitializePackage(Module: HMODULE)
19161: Procedure INITIACTION
19162: Procedure initHexArray(var hexn: THexArray); //THexArray', 'array[0..15] of char,'
19163: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet )
19164: Procedure InitModule( AModule : TComponent )
19165: Procedure InitStdConvs
19166: Procedure InitTreeData( Tree : TUpdateTree )
19167: Procedure INSERT
19168: Procedure Insert( Index : Integer; AClass : TClass )
19169: Procedure Insert( Index : Integer; AComponent : TComponent )
19170: Procedure Insert( Index : Integer; AObject : TObject )
19171: Procedure Insert( Index : Integer; const S : WideString )
19172: Procedure Insert( Index : Integer; Image, Mask : TBitmap )
19173: Procedure Insert(Index: Integer; const S: string);
19174: procedure Insert(Index: Integer; S: string);
19175: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
19176: procedure InsertComponent(AComponent:TComponent)
19177: procedure InsertControl(AControl: TControl);
19178: Procedure InsertIcon( Index : Integer; Image : TIcon )
19179: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor )
19180: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject )
19181: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
19182: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte )
19183: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte )
19184: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte )
19185: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
19186: Procedure InternalBeforeResolve( Tree : TUpdateTree )
19187: Procedure InvalidateModuleCache
19188: Procedure InvalidateTitles
19189: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word )
19190: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word )

```

```

19191: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
19192:   ABaseDate:TDateTime)
19193: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
19194: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
19195: procedure JavaSyntax1Click(Sender : TObject);
19196: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
19197: Procedure KillDataChannel
19198: Procedure Largefont1Click( Sender : TObject)
19199: Procedure LAST
19200: Procedure Launch( const AFile : string)
19201: Procedure LaunchFile( const AFile : string)
19202: Procedure LetFileList(FileList: TStringlist; apath: string);
19203: Procedure lineToNumber( xmemo : String; met : boolean)
19204: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
19205:   DefaultDraw:Bool)
19206: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
19207:   : TCustDrawState; var DefaultDraw : Boolean)
19208: Procedure ListViewData( Sender : TObject; Item : TListItem)
19209: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
19210:   : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
19211: Procedure ListViewHint( Sender : TObject; StartIndex, EndIndex : Integer)
19212: Procedure ListViewDblClick( Sender : TObject)
19213: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
19214: Procedure LoadDLLExports(const FileName: string; List: TStrings);
19215: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
19216: procedure LoadBytecode1Click(Sender: TObject);
19217: procedure LoadFromFileFromResource(const FileName: string; ms: TMemoryStream);
19218: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
19219: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
19220: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
19221: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
19222: Procedure LoadFromFile( const FileName : WideString)
19223: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
19224: Procedure LoadFromFile(const AFileName: string)
19225: procedure LoadFromFile(FileName: string);
19226: procedure LoadFromFile(FileName:String)
19227: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
19228: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
19229: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
19230: Procedure LoadFromStream( const Stream : TStream)
19231: Procedure LoadFromStream( S : TStream)
19232: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
19233: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19234: Procedure LoadFromStream( Stream : TStream)
19235: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
19236: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
19237: procedure LoadFromStream(Stream: TStream);
19238: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
19239: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
19240: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
19241: Procedure LoadLastFile1Click( Sender : TObject)
19242: { LoadIconToImage loads two icons from resource named NameRes,
19243:   into two image lists ALarge and ASmall}
19244: Procedure LoadIconToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
19245: Procedure LoadMemo
19246: Procedure LoadParamsFromIniFile( FFileName : WideString)
19247: Procedure Lock
19248: Procedure Login
19249: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
19250: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
19251: Procedure MakeCaseInsensitive
19252: Procedure MakeDeterministic( var bChanged : boolean)
19253: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
19254: // type TVolumeLevel = 0..127; , saveFilePath as C:\MyFile.wav
19255: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
19256: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:boolean;Volume:Byte);
19257: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
19258: Procedure SetRectComplexFormatStr( const S : string)
19259: Procedure SetPolarComplexFormatStr( const S : string)
19260: Procedure AddComplexSoundObjectToList(newf,newp,newa,newn:integer; freqlist: TStrings);
19261: Procedure MakeVisible
19262: Procedure MakeVisible( PartialOK : Boolean)
19263: Procedure Manual1Click( Sender : TObject)
19264: Procedure MarkReachable
19265: Procedure maxBox; //shows the exe version data in a win box
19266: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
19267: Procedure Memo1Change( Sender : TObject)
19268: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
19269:   Action:TSynReplaceAction)
19270: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
19271: procedure Memory1Click(Sender: TObject);
19272: Procedure MERGE( MENU : TMAINMENU)
19273: Procedure MergeChangeLog
19274: procedure MINIMIZE

```

```

19275: Procedure MinimizeMaxbox;
19276: procedure MyCopyFile(Namel,Name2:string);
19277: Procedure MkDir(const s: string)
19278: Procedure MakeDir(const s: string)');
19279: Procedure ChangeDir(const s: string)');
19280: Function makefile(const FileName: string): integer)';
19281: Procedure mnPrintFont1Click( Sender : TObject)
19282: procedure ModalStarted
19283: Procedure Modified
19284: Procedure ModifyAlias( Name : string; List : TStrings)
19285: Procedure ModifyDriver( Name : string; List : TStrings)
19286: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
19287: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
19288: Procedure Move( CurIndex, NewIndex : Integer)
19289: procedure Move(CurIndex, NewIndex: Integer);
19290: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
19291: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
19292: Procedure moveCube( o : TMyLabel )
19293: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
19294: procedure MoveTo(X, Y: Integer);
19295: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
19296: Procedure MovePoint(var x,y:Extended; const angle:Extended);
19297: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
19298: Procedure Multiply( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
19299: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK );
19300: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
19301: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
19302: procedure New(P: PChar)
19303: procedure NewlClick(Sender: TObject);
19304: procedure NewInstance1Click(Sender: TObject);
19305: Procedure NEXT
19306: Procedure NextMonth
19307: Procedure Noop
19308: Procedure NormalizePath( var APath : string)
19309: procedure ObjectBinaryToText( Input, Output: TStream)
19310: procedure ObjectBinaryToText1( Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19311: procedure ObjectResourceToText( Input, Output: TStream)
19312: procedure ObjectResourceToText1( Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19313: procedure ObjectTextToBinary( Input, Output: TStream)
19314: procedure ObjectTextToBinary1( Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19315: procedure ObjectTextToResource( Input, Output: TStream)
19316: procedure ObjectTextToResource1( Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19317: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
19318: Procedure Open( const UserID : WideString; const Password : WideString);
19319: Procedure Open;
19320: Procedure open1Click( Sender : TObject)
19321: Procedure OpenCdDrive
19322: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
19323: Procedure OpenCurrent
19324: Procedure OpenFile(vfilenamepath: string)
19325: Procedure OpenDirectory1Click( Sender : TObject)
19326: Procedure OpenDir(adir: string);
19327: Procedure OpenIndexFile( const IndexName : string)
19328: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
SchemID:OleVariant;DataSet:TADODataSet)
19329: Procedure OpenWriteBuffer( const AThreshold : Integer)
19330: Procedure OptimizeMem
19331: Procedure Options1( AURL : string);
19332: Procedure OutputDebugString(lpOutputString : PChar)
19333: Procedure PackBuffer
19334: Procedure Paint
19335: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
19336: Procedure PaintToTBitmap( Target : TBitmap)
19337: Procedure PaletteChanged
19338: Procedure ParentBiDiModeChanged
19339: Procedure PARENTBIDI MODECHANGED( ACONTROL : TOBJECT)
19340: Procedure PasteFromClipboard;
19341: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
19342: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
19343: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
19344: Procedure PError( Text : string)
19345: procedure Pie(X1,Y1,X2,Y2,X3,Y3,X4,Y4: Integer);
19346: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
19347: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
19348: procedure playmp3(mpath: string);
19349: Procedure PlayMP31Click( Sender : TObject)
19350: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
19351: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
19352: procedure PolyBezier(const Points: array of TPoint);
19353: procedure PolyBezierTo(const Points: array of TPoint);
19354: procedure Polygon(const Points: array of TPoint);
19355: procedure Polyline(const Points: array of TPoint);
19356: Procedure Pop
19357: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
19358: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
19359: Procedure POPUP( X, Y : INTEGER)
19360: Procedure PopupURL(URL : WideString);
19361: Procedure POST
19362: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);

```

```

19363: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
19364: Procedure Post6( AURL : string; const ASource : TIIdMultiPartFormDataStream; AResponseContent : TStream);
19365: Procedure PostUser( const Email, FirstName, LastName : WideString);
19366: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
19367: procedure Pred(X: int64);
19368: Procedure Prepare;
19369: Procedure PrepareStatement;
19370: Procedure PreProcessXML( AList : TStrings);
19371: Procedure PreventDestruction;
19372: Procedure Print( const Caption : string);
19373: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
19374: procedure printf(const format: String; const args: array of const);
19375: Procedure PrintList(Value: TStringList);
19376: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle); //TBitmapStyle=(bsNormal,bsCentered,bsStretched)
19377: Procedure Printout1Click( Sender : TObject);
19378: Procedure ProcessHeaders;
19379: Procedure PROCESSENCHAR( var MESSAGE : TWMMENUCHAR);
19380: Procedure ProcessMessage( AMsg : TIIdMessage; AHeaderOnly : Boolean);
19381: Procedure ProcessMessage1( AMsg : TIIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
19382: Procedure ProcessMessage2( AMsg : TIIdMessage; const AFilename : string; AHeaderOnly : Boolean);
19383: Procedure ProcessMessagesOFF; //application.processmessages
19384: Procedure ProcessMessagesON;
19385: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string);
19386: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
19387: Procedure Proclist Size is: 3797 /1415
19388: Procedure procMessClick( Sender : TObject);
19389: Procedure PSScriptCompile( Sender : TPSScript);
19390: Procedure PSScriptExecute( Sender : TPSScript);
19391: Procedure PSScriptLine( Sender : TObject);
19392: Procedure Push( ABoundary : string);
19393: procedure PushItem(AItem: Pointer);
19394: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
19395: Procedure Put2( const ASourcefile : string; const ADestFile : string; const AAppend : boolean);
19396: procedure PutLinuxLines(const Value: string);
19397: Procedure Quit;
19398: Procedure RaiseConversionError( const AText : string);
19399: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
19400: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string);
19401: procedure RaiseException(Ex: TIEException; Param: String);
19402: Procedure RaiseExceptionForLastCmdResult;
19403: procedure RaiseLastException;
19404: procedure RaiseException2;
19405: Procedure RaiseException3(const Msg: string);
19406: Procedure RaiseExcept(const Msg: string);
19407: Procedure RaiseLastOSError;
19408: Procedure RaiseLastWin32;
19409: procedure RaiseLastWin32Error;
19410: Procedure RaiseListError( const ATemplate : string; const AData : array of const);
19411: Procedure RandomFillStream( Stream : TMemoryStream);
19412: procedure randomize;
19413: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect);
19414: Procedure RCS;
19415: Procedure Read( Socket : TSocket);
19416: procedure ReadInl(var ast: string); //of inputquery;
19417: Procedure ReadBlobData;
19418: procedure ReadBuffer(Buffer:String;Count:LongInt);
19419: procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2,readonly:= false;
19420: Procedure ReadSection( const Section : string; Strings : TStrings);
19421: Procedure ReadSections( Strings : TStrings);
19422: Procedure ReadSections( Strings : TStrings);
19423: Procedure ReadSections1( const Section : string; Strings : TStrings);
19424: Procedure ReadSectionValues( const Section : string; Strings : TStrings);
19425: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean);
19426: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer);
19427: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
19428: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
19429: Procedure Realign;
19430: procedure Rectangle(X1, Y1, X2, Y2: Integer);
19431: Procedure Rectangle1( const Rect : TRect);
19432: Procedure RectCopy( var Dest : TRect; const Source : TRect);
19433: Procedure RectFitToScreen( var R : TRect);
19434: Procedure RectGrow( var R : TRect; const Delta : Integer);
19435: Procedure RectGrowX( var R : TRect; const Delta : Integer);
19436: Procedure RectGrowY( var R : TRect; const Delta : Integer);
19437: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer);
19438: Procedure RectMoveTo( var R : TRect; const X, Y : Integer);
19439: Procedure RectNormalize( var R : TRect);
19440: // TFileCallbackProcedure = procedure(filename:string);
19441: Procedure RecurseDirectory( Dir: String; IncludeSubs: boolean; callback: TFileCallbackProcedure);
19442: Procedure RecurseDirectory2( Dir: String; IncludeSubs : boolean);
19443: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState);
19444: Procedure Refresh;
19445: Procedure RefreshData( Options : TFetchOptions);
19446: Procedure REFRESHLOOKUPLIST;
19447: Procedure regExPathfinder(Pathin, fileout, firsttp, aregex, ext: string; asort: boolean);
19448: Procedure RegExPathfinder2(Pathin, fileout, firsttp, aregex, ext: string; asort, acopy: boolean);
19449: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIIdAuthenticationClass);
19450: Procedure RegisterChanges( Value : TChangeLink);
19451: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass);

```

```

19452: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass )
19453: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
19454: Procedure ReInitialize( ADelay : Cardinal )
19455: procedure RELEASE
19456: Procedure Remove( const AByteCount : integer )
19457: Procedure REMOVE( FIELD : TFIELD )
19458: Procedure REMOVE( ITEM : TMENUITEM )
19459: Procedure REMOVE( POPUP : TPOPUPMENU )
19460: Procedure RemoveAllPasswords
19461: procedure RemoveComponent( AComponent : TComponent )
19462: Procedure RemoveDir( const ADirName : string )
19463: Procedure RemoveLambdaTransitions( var bChanged : boolean )
19464: Procedure REMOVEPARAM( VALUE : TPARAM )
19465: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset );
19466: Procedure RemoveTransitionTo( oState : TniRegularExpressionState );
19467: Procedure Rename( const ASourceFile, ADestFile : string )
19468: Procedure Rename( const FileName : string; Reload : Boolean )
19469: Procedure RenameTable( const NewTableName : string )
19470: Procedure Replace( Index : Integer; Image, Mask : TBitmap )
19471: Procedure ReplaceClick( Sender : TObject )
19472: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime )
19473: procedure ReplaceDate( var DateTime : TDateTime; const NewDate : TDateTime ) )
19474: Procedure ReplaceIcon( Index : Integer; Image : TIcon )
19475: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor )
19476: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime )
19477: procedure ReplaceTime( var DateTime : TDateTime; const NewTime : TDateTime );
19478: Procedure Requery( Options : TExecuteOptions )
19479: Procedure Reset
19480: Procedure Reset1Click( Sender : TObject )
19481: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor )
19482: procedure ResourceExplore1Click(Sender: TObject);
19483: Procedure RestoreContents
19484: Procedure RestoreDefaults
19485: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string )
19486: Procedure RetrieveHeaders
19487: Procedure RevertRecord
19488: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal )
19489: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal );
19490: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal );
19491: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single );
19492: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single );
19493: Procedure RGBtoHSV( r, g, b : Integer; var h, s, v : Integer )
19494: Procedure RleCompress2( Stream : TStream )
19495: Procedure RleDecompress2( Stream : TStream )
19496: Procedure RmDir( const S : string )
19497: Procedure Rollback
19498: Procedure Rollback( TransDesc : TTransactionDesc )
19499: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction )
19500: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction )
19501: Procedure RollbackTrans
19502: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
19503: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean )
19504: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean )
19505: Procedure RunD132Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer )
19506: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string )
19507: Procedure S_EBox( const AText : string )
19508: Procedure S_GetEncryptionKeys( DateTimel, DTime2: TDateTime; var StartKey:int; var MultKey:int; var AddKey:int )
19509: Procedure S_IBox( const AText : string )
19510: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char )
19511: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string )
19512: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string )
19513: Procedure SampleVarianceAndMean
19514: ( const X : TDynFloatArray; var Variance, Mean : Float )
19515: Procedure Save2Click( Sender : TObject )
19516: Procedure Saveas3Click( Sender : TObject )
19517: Procedure Savebefore1Click( Sender : TObject )
19518: Procedure SaveBytesToFile( const Data : TBytes; const FileName: string );
19519: procedure SaveCanvas2( vCanvas: TCanvas; FileName: string );
19520: Procedure SaveConfigFile
19521: Procedure SaveOutput1Click( Sender : TObject )
19522: procedure SaveScreenshotClick(Sender: TObject);
19523: Procedure SaveIn(pathname, content: string); //Saveln(exepath+'mysavelntest.txt', memo2.text);
19524: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APALETTE : HPALETTE )
19525: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APALETTE : HPALETTE )
19526: Procedure SaveToFile( AFileName : string )
19527: Procedure SAVETOFILE( const FILENAME : String )
19528: Procedure SaveToFile( const FileName : WideString )
19529: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat )
19530: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap )
19531: procedure SaveToFile(FileName: string);
19532: procedure SaveToFile(FileName:String)
19533: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean )
19534: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap )
19535: Procedure SaveToStream( S : TStream )
19536: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap )
19537: Procedure SaveToStream( Stream : TStream )
19538: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat )
19539: procedure SaveToStream(Stream: TStream);
19540: procedure SaveToStream(Stream:TStream)

```

```

19541: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
19542: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
19543: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
19544: procedure Say(const sText: string)
19545: Procedure SBytecode1Click( Sender : TObject)
19546: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
19547: procedure ScriptExplorer1Click(Sender: TObject);
19548: Procedure Scroll( Distance : Integer)
19549: Procedure Scroll( DX, DY : Integer)
19550: procedure ScrollBy(DeltaX, DeltaY: Integer);
19551: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
19552: Procedure ScrollTabs( Delta : Integer)
19553: Procedure Search1Click( Sender : TObject)
19554: procedure SearchAndOpenDoc(vfilenamepath: string)
19555: procedure SearchAndOpenFile(vfilenamepath: string)
19556: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
19557: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19558: Procedure SearchNext1Click( Sender : TObject)
19559: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
19560: Procedure Select1( const Nodes : array of TTreeNode);
19561: Procedure Select2( Nodes : TList);
19562: Procedure SelectNext( Direction : Boolean)
19563: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
19564: Procedure SelfTestPEM //unit uPSI_CPEM
19565: Procedure Send( AMsg : TIdMessage)
19566: //config forst in const MAILINITFILE = 'maildef.ini';
19567: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
19568: Procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
19569: Procedure SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
19570: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
19571: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
19572: Procedure SendResponse
19573: Procedure SendStream( AStream : TStream)
19574: procedure SendMCICmd(Cmd: string); !
19575: Procedure Set8087CW( NewCW : Word)
19576: Procedure SetAll( One, Two, Three, Four : Byte)
19577: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
19578: Procedure SetAppDispatcher( const AD dispatcher : TComponent)
19579: procedure SetArrayLength;
19580: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer); //2 dimension
19581: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19582: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer); //all init
19583: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
19584: procedure SetArrayLength2Char2(var arr: T2CharArray; asize1, asize2: integer);'
19585: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);'
19586: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);'
19587: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
19588: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
19589: Procedure SetAsHandle( Format : Word; Value : THandle)
19590: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
19591: procedure SetCaptureControl(Control: TControl);
19592: Procedure SetColumnAttributes
19593: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
19594: Procedure SetCustomHeader( const Name, Value : string)
19595: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
  FieldName:Widestring)
19596: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
19597: Procedure SetFocus
19598: procedure SetFocus; virtual;
19599: Procedure SetInitialState
19600: Procedure SetKey
19601: procedure SetLastError(ErrorCode: Integer)
19602: procedure SetLength;
19603: procedure SetLength2(var S: string; NewLength: Integer)');
19604: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
19605: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
19606: Procedure SetParams( ADataSet : TDataSet; UpdateKind : TUpdateKind);
19607: procedure SETPARAMS(APosition,AMIN,AMAX:INTEGER)
19608: Procedure SetParams1( UpdateKind : TUpdateKind);
19609: Procedure SetPassword( const Password : string)
19610: Procedure SetPointer( Ptr : Pointer; Size : Longint)
19611: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
19612: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
19613: Procedure SetProvider( Provider : TComponent)
19614: Procedure SetProxy( const Proxy : string)
19615: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
19616: Procedure SetRange( const StartValues, EndValues : array of const)
19617: Procedure SetRangeEnd
19618: Procedure SetRate( const aPercent, aYear : integer)
19619: procedure SetRate(const aPercent, aYear: integer)
19620: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19621: Procedure SetSafeCallExceptionMsg( Msg : String)
19622: procedure SETSELTEXTBUF(BUFFER:PCHAR)
19623: Procedure SetSize( AWidth, AHeight : Integer)
19624: procedure SetSize(NewSize:LongInt)
19625: procedure SetString(var s: string; buffer: PChar; len: Integer)
19626: Procedure SetStrings( List : TStrings)
19627: Procedure SetText( Text : PwideChar)
19628: procedure SetText(Text: PChar);

```

```

19629: Procedure SetTextBuf( Buffer : PChar)
19630: procedure SETTEXTBUF(BUFFER:PCCHAR)
19631: Procedure SetTick( Value : Integer)
19632: Procedure SetTimeout( ATimeOut : Integer)
19633: Procedure SetTraceEvent( Event : TDBXTraceEvent)
19634: Procedure SetUserName( const UserName : string)
19635: Procedure SetWallpaper( Path : string);
19636: procedure ShellStyle1Click(Sender: TObject);
19637: Procedure SHORTCUTTOKEY( SHORCUT : TSHORCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
19638: Procedure ShowFileProperties( const FileName : string)
19639: Procedure ShowInclude1Click( Sender : TObject)
19640: Procedure ShowInterfaces1Click( Sender : TObject)
19641: Procedure ShowLastException1Click( Sender : TObject)
19642: Procedure ShowMessage( const Msg : string)
19643: Procedure ShowMessageBig(const aText : string);
19644: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
19645: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
19646: Procedure MsgBig(const aText : string); //alias
19647: procedure showmessage(mytext: string);
19648: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
19649: procedure ShowMessageFmt(const Msg: string; Params: array of const))
19650: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
19651: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
19652: Procedure ShowSearchDialog( const Directory : string)
19653: Procedure ShowSpecChars1Click( Sender : TObject)
19654: Procedure ShowBitmap(bmap: TBitmap); //draw in a form!
19655: Procedure ShredFile( const FileName : string; Times : Integer)
19656: procedure Shuffle(vQ: TStringList);
19657: Procedure ShuffleList( var List : array of Integer; Count : Integer)
19658: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
19659: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
19660: Procedure SinCose( X : Extended; out Sin, Cos : Extended)
19661: Procedure Site( const ACommand : string)
19662: Procedure SkipEOL
19663: Procedure Sleep( ATime : cardinal)
19664: Procedure Sleep( milliseconds : Cardinal)
19665: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
19666: Procedure Slinenumbers1Click( Sender : TObject)
19667: Procedure Sort
19668: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
19669: procedure SoundAlarm; //beep seq
19670: procedure Speak(const sText: string) //async like voice
19671: procedure Speak2(const sText: string) //sync
19672: procedure Split(Str: string; SubStr: string; List: TStrings);
19673: Procedure SplitNameValuePair( const Line : string; var Name, Value : string)
19674: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
19675: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
19676: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
19677: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
19678: procedure SQLSyntax1Click(Sender: TObject);
19679: Procedure SRand( Seed : RNG_IntType)
19680: Procedure Start
19681: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
19682: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
19683: //Ex. StartFileFinder3(exepath+'exercises','*.pas','record',false,seclist);
19684: Procedure StartTransaction( TransDesc : TTransactionDesc)
19685: Procedure Status( var AStatusList : TStringList)
19686: Procedure StatusBar1DblClick( Sender : TObject)
19687: Procedure StepInto1Click( Sender : TObject)
19688: Procedure StepIt
19689: Procedure StepOut1Click( Sender : TObject)
19690: Procedure Stop
19691: procedure stopmp3;
19692: procedure StartWeb(aurl: string);
19693: Procedure Str(aint: integer; astr: string); //of system
19694: Procedure StrDispose( Str : PChar)
19695: procedure StrDispose(Str: PChar)
19696: Procedure StrReplace(var Str: String; Old, New: String);
19697: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
19698: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
19699: Procedure StringToBytes( Value : String; Bytes : array of byte)
19700: procedure StrSet(c : Char; I : Integer; var s : String);
19701: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19702: Procedure StructureMount( APath : String)
19703: procedure STYLECHANGED(SENDER:TOBJECT)
19704: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
19705: procedure Succ(X: int64);
19706: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
19707: procedure SwapChar(var X,Y: char); //swapX follows
19708: Procedure SwapFloats( var X, Y : Float)
19709: procedure SwapGrid(grd: TStringGrid);
19710: Procedure SwapOrd( var I, J : Byte);
19711: Procedure SwapOrd( var X, Y : Integer)
19712: Procedure SwapOrd1( var I, J : Shortint);
19713: Procedure SwapOrd2( var I, J : Smallint);
19714: Procedure SwapOrd3( var I, J : Word);
19715: Procedure SwapOrd4( var I, J : Integer);
19716: Procedure SwapOrd5( var I, J : Cardinal);
19717: Procedure SwapOrd6( var I, J : Int64);

```

```

19718: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
19719: Procedure Synchronizel( Method : TMethod);
19720: procedure SyntaxCheck1Click(Sender: TObject);
19721: Procedure SysFreeString(const S: WideString); stdcall;
19722: Procedure TakeOver( Other : TLinearBitmap)
19723: Procedure TalkIn(const sText: string) //async voice
19724: procedure tbtn6resClick(Sender: TObject);
19725: Procedure tbtmUseCaseClick( Sender : TObject)
19726: procedure TerminalStyle1Click(Sender: TObject);
19727: Procedure Terminate
19728: Procedure texSyntax1Click( Sender : TObject)
19729: procedure TextOut(X, Y: Integer; Text: string);
19730: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
19731: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
19732: Procedure TextRectl( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
19733: Procedure TextStart
19734: procedure TILE
19735: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
19736: Procedure TitleClick( Column : TColumn)
19737: Procedure ToDo
19738: procedure Tone(500 hz, 10000 ms length);
19739: procedure toolbtnTutorialClick(Sender: TObject);
19740: Procedure Trace1( AURL : string; const AResponseContent : TStream);
19741: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
19742: Procedure Truncate
19743: procedure Tutorial101Click(Sender: TObject);
19744: procedure Tutorial10Statistics1Click(Sender: TObject);
19745: procedure Tutorial11Forms1Click(Sender: TObject);
19746: procedure Tutorial12SQL1Click(Sender: TObject);
19747: Procedure tutorial1Click( Sender : TObject)
19748: Procedure tutorial21Click( Sender : TObject)
19749: Procedure tutorial31Click( Sender : TObject)
19750: Procedure tutorial4Click( Sender : TObject)
19751: Procedure Tutorial5Click( Sender : TObject)
19752: procedure Tutorial6Click(Sender: TObject);
19753: procedure Tutorial91Click(Sender: TObject);
19754: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
19755: procedure UniqueString(var str: AnsiString)
19756: procedure UploadLoadPackage(Module: HMODULE)
19757: Procedure Unlock
19758: Procedure UNMERGE( MENU : TMainMenu)
19759: Procedure UnRegisterChanges( Value : TChangeLink)
19760: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
19761: Procedure UnregisterConversionType( const AType : TConvType)
19762: Procedure UnRegisterProvider( Prov : TCustomProvider)
19763: Procedure UPDATE
19764: Procedure UpdateBatch( AffectRecords : TAffectRecords)
19765: Procedure UPDATECURSORPOS
19766: Procedure UpdateFile
19767: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
19768: Procedure UpdateResponse( AResponse : TWebResponse)
19769: Procedure UpdateScrollBar
19770: Procedure UpdateView1Click( Sender : TObject)
19771: procedure UpdateExeResource(Const Source,Dest:string); //!
19772: procedure Val(const s: string; var n, z: Integer)
19773: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
19774: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
19775: Procedure VariantAdd( const src : Variant; var dst : Variant)
19776: Procedure VariantAnd( const src : Variant; var dst : Variant)
19777: Procedure VariantArrayRedim( var V : Variant; High : Integer)
19778: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
19779: Procedure VariantClear( var V : Variant)
19780: Procedure VariantCpy( const src : Variant; var dst : Variant)
19781: Procedure VariantDiv( const src : Variant; var dst : Variant)
19782: Procedure VariantMod( const src : Variant; var dst : Variant)
19783: Procedure VariantMul( const src : Variant; var dst : Variant)
19784: Procedure VariantOr( const src : Variant; var dst : Variant)
19785: Procedure VariantPutElement( var V : Variant; const data : Variant; il : integer);
19786: Procedure VariantPutElement1( var V : Variant; const data : Variant; il, i2 : integer);
19787: Procedure VariantPutElement2( var V : Variant; const data : Variant; il, i2, i3 : integer);
19788: Procedure VariantPutElement3( var V : Variant; const data : Variant; il, i2, i3, i4 : integer);
19789: Procedure VariantPutElement4( var V : Variant; const data : Variant; il, i2, i3, i4, i5 : integer);
19790: Procedure VariantShl( const src : Variant; var dst : Variant)
19791: Procedure VariantShr( const src : Variant; var dst : Variant)
19792: Procedure VariantSub( const src : Variant; var dst : Variant)
19793: Procedure VariantXor( const src : Variant; var dst : Variant)
19794: Procedure VarCastError;
19795: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
19796: Procedure VarInvalidOp
19797: Procedure VarInvalidNullOp
19798: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
19799: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
19800: Procedure VarArrayCreateError
19801: Procedure VarResultCheck( AResult : HRESULT);
19802: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
19803: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
19804: Function VarTypeAsText( const AType : TVarType) : string
19805: procedure Voice(const sText: string) //async
19806: procedure Voice2(const sText: string) //sync

```

```

19807: Procedure WaitMiliSeconds( AMSec : word)
19808: Procedure WaitMS( AMSec : word)'');
19809: procedure WebCampic(picname: string); //eg: c:\mypic.png
19810: Procedure WideAppend( var dst : WideString; const src : WideString)
19811: Procedure WideAssign( var dst : WideString; var src : WideString)
19812: Procedure WideDelete( var dst : WideString; index, count : Integer)
19813: Procedure WideFree( var s : WideString)
19814: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
19815: Procedure WideFromPChar( var dst : WideString; src : PChar)
19816: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
19817: Procedure WideSetLength( var dst : WideString; len : Integer)
19818: Procedure Widestring2Stream( aWideString : WideString; oStream : TStream)
19819: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
19820: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
19821: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19822: Procedure HttpGet(const Url: string; Stream:TStream);
19823: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIddBytes; Index : integer)
19824: Procedure WordWrap1Click( Sender : TObject)
19825: Procedure Write( const AOut : string)
19826: Procedure Write( Socket : TSocket)
19827: procedure Write(S: string);
19828: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
19829: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
19830: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
19831: procedure WriteBuffer(Buffer:String;Count:LongInt)
19832: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
19833: Procedure WriteChar( AValue : Char)
19834: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
19835: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
19836: Procedure WriteFloat( const Section, Name : string; Value : Double)
19837: Procedure WriteHeader( AHeader : TStrings)
19838: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
19839: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
19840: Procedure WriteLn( const AOut : string)
19841: procedure Writeln(s: string);
19842: Procedure WriteLog( const FileName, LogLine : string)
19843: Procedure WriteRFCReply( AReply : TIIdRFCReply)
19844: Procedure WriteRFCStrings( AStrings : TStrings)
19845: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
19846: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
19847: Procedure WriteString( const Section, Ident, Value : String)
19848: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
19849: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
19850: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
19851: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19852: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19853: Procedure WriteDataSetToCSV(DataSet : TDataSet; FileName: String)');
19854: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
19855: procedure XMLSyntax1Click(Sender: TObject);
19856: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
19857: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
19858: Procedure ZeroFillStream( Stream : TMemoryStream)
19859: procedure XMLSyntax1Click(Sender: TObject);
19860: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
19861: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
19862: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
19863: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
19864: procedure(Sender, Source: TObject; X, Y: Integer)
19865: procedure(Sender, Target: TObject; X, Y: Integer)
19866: procedure(Sender: TObject; ASection, AWidth: Integer)
19867: procedure(Sender: TObject; ScrollCode: TScrollCode; var ScrollPos: Integer)
19868: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer)
19869: procedure(Sender: TObject; var Action: TCcloseAction)
19870: procedure(Sender: TObject; var CanClose: Boolean)
19871: procedure(Sender: TObject; var Key: Char);
19872: ProcedureName ProcedureNames ProcedureParametersCursor @
19873:
19874: *****Now Constructors constructor *****
19875: Size is: 1248 1115 996 628 550 544 501 459 (381)
19876: Attach( VersionInfoData : Pointer; Size : Integer)
19877: constructor Create( ABuckets : TBucketListSizes)
19878: Create( ACallBackWnd : Hwnd)
19879: Create( AClient : TCustomTaskDialog)
19880: Create( AClient : TIIdTelnet)
19881: Create( ACollection : TCollection)
19882: Create( ACollection : TFavoriteLinkItems)
19883: Create( ACollection : TTaskDialogButtons)
19884: Create( AConnection : TIIdCustomHTTP)
19885: Create( ACreatSuspended : Boolean)
19886: Create( ADataSet : TCustomSQLDataSet)
19887: CREATE( ADATASET : TDATASET)
19888: Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
19889: Create( AGrid : TCustomDBGrid)
19890: Create( AGrid : TStringGrid; AIndex : Longint)
19891: Create( AHTTP : TIIdCustomHTTP)
19892: Create( AListItems : TListItems)
19893: Create( AOnBytesRemoved : TIIdBufferBytesRemoved)
19894: Create( AOnBytesRemoved : TIIdBufferBytesRemoved)
19895: Create( AOwner : TCommonCalendar)

```

```

19896: Create( AOwner : TComponent )
19897: CREATE( AOWNER : TCOMPONENT )
19898: Create( AOwner : TCustomListView )
19899: Create( AOwner : TCustomOutline )
19900: Create( AOwner : TCustomRichEdit )
19901: Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType )
19902: Create( AOwner : TCustomTreeView )
19903: Create( AOwner : TIdUserManager )
19904: Create( AOwner : TListItems )
19905: Create(AOwner:TObj;Handle:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
19906: CREATE( AOWNER : TPERSISTENT )
19907: Create( AOwner : TPersistent )
19908: Create( AOwner : TTable )
19909: Create( AOwner : TTreeNodes )
19910: Create( AOwner : TWinControl; const ClassName : string )
19911: Create( AParent : TIdCustomHTTP )
19912: Create( AParent : TUpdateTree; AResolver : TCustomResolver )
19913: Create( AProvider : TBaseProvider )
19914: Create( AProvider : TCustomProvider );
19915: Create( AProvider : TDataSetProvider )
19916: Create( ASocket : TCustomWinSocket; TimeOut : Longint )
19917: Create( ASocket : TSocket )
19918: Create( AStrings : TWideStrings )
19919: Create( AToolBar : TToolBar )
19920: Create( ATreeNodes : TTreeNodes )
19921: Create( Autofill : boolean )
19922: Create( AWebPageInfo : TABstractWebPageInfo )
19923: Create( AWebRequest : TWebRequest )
19924: Create( Collection : TCollection )
19925: Create( Collection : TIdMessageParts; ABody : TStrings )
19926: Create( Collection : TIdMessageParts; const AFileName : TFileName )
19927: Create( Column : TColumn )
19928: Create( const AConvFamily : TConvFamily; const ADescription : string )
19929: Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double )
19930: Create( const AConvFamily:TConvFamily;const ADescription:string;const AToCommonProc,
AFromComProc:TConversionProc )
19931: Create( const AInitialState : Boolean; const AManualReset : Boolean )
19932: Create( const ATabSet : TTabSet )
19933: Create( const Compensate : Boolean )
19934: Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64 )
19935: Create( const FileName : string )
19936: Create( const FileName : string; FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
: Int64; const SecAttr : PSecurityAttributes );
19937: Create( const FileName : string; FileMode : WordfmShareDenyWrite )
19938: Create( const MaskValue : string )
19939: Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
19940: Create( const Prefix : string )
19941: Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags )
19942: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags )
19943: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags )
19944: Create( CoolBar : TCoolBar )
19945: Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket )
19946: Create( CreateSuspended : Boolean; ASocket : TServerWinSocket )
19947: Create( DataSet : TDataSet; const
Text:WideString;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : WideString;
DepFields : TBits; FieldMap : TFieldMap )
19948: Create( DBCtrlGrid : TDBCtrlGrid )
19949: Create( DSTableProducer : TDSTableProducer )
19950: Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass )
19951: Create( ErrorCode : DBIResult )
19952: Create( Field : TblobField; Mode : TblobStreamMode )
19953: Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass )
19954: Create( HeaderControl : TCustomHeaderControl )
19955: Create( HTTPRequest : TWebRequest )
19956: Create( iStart : integer; sText : string )
19957: Create( iValue : Integer )
19958: Create( Kind : TMmTimerKind; Notification : TMmNotificationKind )
19959: Create( MciErrNo : MCIERRO; const Msg : string )
19960: Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
19961: Create( Message : string; ErrorCode : DBResult )
19962: Create( Msg : string )
19963: Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception )
19964: Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult )
19965: Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType )
19966: Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags )
19967: Create(oSource:TniRegularExpressionState;oDestination:TniRegularExpressionState;xCharacts:TCharSet;bLambda:bool)
19968: Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar )
19969: Create( Owner : TCustomComboBoxEx )
19970: CREATE( OWNER : TINDEXDEFNS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS )
19971: Create( Owner : TPersistent )
19972: Create( Params : TStrings ) Create( Size : Cardinal )
19973: Create( Socket : TSocket; ServerWinSocket : TServerWinSocket )
19974: Create( StatusBar : TCustomStatusBar )
19975: Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass )
19976: Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass )
19977: Create(AHandle:Integer)
19978: Create(AOwner: TComponent); virtual;
19979: Create(const AURI : string)
19980: Create(Filename:String;Mode:Word)

```

```

19981: Create(Instance:THandle;ResName:String;ResType:PChar)
19982: Create(Stream : TStream)
19983: CreateDataset(ADataset : TDataset);
19984: CreateDataset(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes);
19985: CreateDataset( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
19986: Create2( Other : TObject);
19987: CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
19988: CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
19989: CreateFmt( MciErrNo : MCIERRO; const Msg : string; const Args : array of const)
19990: CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
19991: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
19992: CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
19993: CreateRes( Ident : Integer);
19994: CreateRes( MciErrNo : MCIERRO; Ident : Integer)
19995: CreateRes( ResStringRec : PResStringRec);
19996: CreateResHelp( Ident : Integer; AHelpContext : Integer);
19997: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
19998: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
19999: CreateSize( AWidth, AHeight : Integer)
20000: Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
20001:
20002: -----
20003: unit UPSI_MathMax;
20004: -----
20005: CONSTS
20006: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
20007: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
20008: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
20009: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
20010: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
20011: CbrtPi: Float = 1.4645918875615232630201425272638; // CubeRoot(PI)
20012: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant
20013: PiJ: Float = 3.1415926535897932384626433832795; // PI
20014: PI: Extended = 3.1415926535897932384626433832795;
20015: PiOn2: Float = 1.5707963267948966192313216916398; // PI / 2
20016: PiOn3: Float = 1.0471975511965977461542144610932; // PI / 3
20017: PiOn4: Float = 0.78539816339744830961566084581988; // PI / 4
20018: Sqrt2: Float = 1.4142135623730950488016887242097; // Sqrt(2)
20019: Sqrt3: Float = 1.7320508075688772935274463415059; // Sqrt(3)
20020: Sqrt5: Float = 2.2360679774997896964091736687313; // Sqrt(5)
20021: Sqrt10: Float = 3.1622776601683793319988935444327; // Sqrt(10)
20022: SqrtPi: Float = 1.7724538509055160272981674833411; // Sqrt(PI)
20023: Sqrt2Pi: Float = 2.506628274631000502415765284811; // Sqrt(2 * PI)
20024: TwoPi: Float = 6.283185307179586476925286766559; // 2 * PI
20025: ThreePi: Float = 9.4247779607693797153879301498385; // 3 * PI
20026: Ln2: Float = 0.69314718055994530941723212145818; // Ln(2)
20027: Ln10: Float = 2.3025850929940456840179914546844; // Ln(10)
20028: LnPi: Float = 1.1447298858494001741434273513531; // Ln(PI)
20029: Log2J: Float = 0.30102999566398119521373889472449; // Log10(2)
20030: Log3: Float = 0.47712125471966243729502790325512; // Log10(3)
20031: LogPi: Float = 0.4971498726941338543512682882909; // Log10(PI)
20032: LogE: Float = 0.43429448190325182765112891891661; // Log10(E)
20033: E: Float = 2.7182818284590452353602874713527; // Natural constant
20034: hLn2Pi: Float = 0.91893853320467274178032973640562; // Ln(2*PI)/2
20035: inv2Pi: Float = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
20036: TwoToPower63: Float = 9223372036854775808.0; // 2^63
20037: GoldenMean: Float = 1.618033988749894848204586834365638; // GoldenMean
20038: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
20039: RadCor: Double = 57.29577951308232; {number of degrees in a radian}
20040: StDelta : Extended = 0.00001; {delta for difference equations}
20041: StEpsilon : Extended = 0.00001; {epsilon for difference equations}
20042: StMaxIterations : Integer = 100; {max attempts for convergence}
20043:
20044: procedure SIRegister_StdConvs(CL: TPSPPascalCompiler);
20045: begin
20046:   MetersPerInch = 0.0254; // [1]
20047:   MetersPerFoot = MetersPerInch * 12;
20048:   MetersPerYard = MetersPerFoot * 3;
20049:   MetersPerMile = MetersPerFoot * 5280;
20050:   MetersPerNauticalMiles = 1852;
20051:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
20052:   MetersPerLightSecond = 2.99792458E8; // [5]
20053:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
20054:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
20055:   MetersPerCubit = 0.4572; // [6][7]
20056:   MetersPerFathom = MetersPerFoot * 6;
20057:   MetersPerFurlong = MetersPerYard * 220;
20058:   MetersPerHand = MetersPerInch * 4;
20059:   MetersPerPace = MetersPerInch * 30;
20060:   MetersPerRod = MetersPerFoot * 16.5;
20061:   MetersPerChain = MetersPerRod * 4;
20062:   MetersPerLink = MetersPerChain / 100;
20063:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
20064:   MetersPerPica = MetersPerPoint * 12;
20065:
20066:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
20067:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
20068:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;

```

```

20069:     SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
20070:     SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
20071:     SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
20072:
20073:     CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
20074:     CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
20075:     CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
20076:     CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
20077:     CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
20078:     CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
20079:     CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
20080:     CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
20081:
20082:     CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
20083:     CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
20084:     CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
20085:     CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
20086:     CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
20087:     CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
20088:     CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
20089:     CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
20090:     CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
20091:     CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
20092:     CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
20093:     CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
20094:     CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
20095:     CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
20096:
20097:     CubicMetersPerUKGallon = 0.00454609; // [2][7]
20098:     CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
20099:     CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
20100:     CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
20101:     CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
20102:     CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
20103:     CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
20104:     CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
20105:     CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
20106:
20107:     GramsPerPound = 453.59237; // [1][7]
20108:     GramsPerDrams = GramsPerPound / 256;
20109:     GramsPerGrains = GramsPerPound / 7000;
20110:     GramsPerTons = GramsPerPound * 2000;
20111:     GramsPerLongTons = GramsPerPound * 2240;
20112:     GramsPerOunces = GramsPerPound / 16;
20113:     GramsPerStones = GramsPerPound * 14;
20114:
20115:     MaxAngle 9223372036854775808.0;
20116:     MaxTanH 5678.261703147071974745965389854);
20117:     MaxFactorial( 1754);
20118:     MaxFloatingPoint(1.189731495357231765085759326628E+4932);
20119:     MinFloatingPoint'(3.3621031431120935062626778173218E-4932);
20120:     MaxTanH( 354.89135644669199842162284618659);
20121:     MaxFactorial'LongInt'( 170);
20122:     MaxFloatingPointD(1.797693134862315907729305190789E+308);
20123:     MinFloatingPointD(2.2250738585072013830902327173324E-308);
20124:     MaxTanH( 44.361419555836499802702855773323);
20125:     MaxFactorial'LongInt'( 33);
20126:     MaxFloatingPoints( 3.4028236692093846346337460743177E+38);
20127:     MinFloatingPoints( 1.1754943508222875079687365372222E-38);
20128:     PiExt( 3.1415926535897932384626433832795);
20129:     RatioDegToRad( PiExt / 180.0);
20130:     RatioGradToRad( PiExt / 200.0);
20131:     RatioDegToGrad( 200.0 / 180.0);
20132:     RatioGradToDeg( 180.0 / 200.0);
20133:     Crc16PolynomCCITT'LongWord $1021);
20134:     Crc16PolynomIBM'LongWord $8005);
20135:     Crc16Bits'LongInt'( 16);
20136:     Crc16Bytes'LongInt'( 2);
20137:     Crc16HighBit'LongWord $8000);
20138:     NotCrc16HighBit', 'LongWord $7FFF);
20139:     Crc32PolynomIEEE', 'LongWord $04C11DB7);
20140:     Crc32PolynomCastagnoli', 'LongWord $1EDC6F41);
20141:     Crc32Koopman', 'LongWord $741B8CD7);
20142:     Crc32Bits', 'LongInt'( 32);
20143:     Crc32Bytes', 'LongInt'( 4);
20144:     Crc32HighBit', 'LongWord $80000000);
20145:     NotCrc32HighBit', 'LongWord $7FFFFFFF);
20146:
20147:     MinByte      = Low(Byte);
20148:     MaxByte      = High(Byte);
20149:     MinWord      = Low(Word);
20150:     MaxWord      = High(Word);
20151:     MinShortInt  = Low(ShortInt);
20152:     MaxShortInt  = High(ShortInt);
20153:     MinSmallInt  = Low(SmallInt);
20154:     MaxSmallInt  = High(SmallInt);
20155:     MinLongWord   = LongWord(Low(LongWord));
20156:     MaxLongWord   = LongWord(High(LongWord));
20157:     MinLongInt   = LongInt(Low(LongInt));

```

```

20158: MaxLongInt    = LongInt(High(LongInt));
20159: MinInt64      = Int64(Low(Int64));
20160: MaxInt64      = Int64(High(Int64));
20161: MinInteger    = Integer(Low(Integer));
20162: MaxInteger    = Integer(High(Integer));
20163: MinCardinal   = Cardinal(Low(Cardinal));
20164: MaxCardinal   = Cardinal(High(Cardinal));
20165: MinNativeUInt = NativeUInt(Low(NativeUInt));
20166: MaxNativeUInt = NativeUInt(High(NativeUInt));
20167: MinNativeInt  = NativeInt(Low(NativeInt));
20168: MaxNativeInt  = NativeInt(High(NativeInt));
20169: Function CosH( const Z : Float) : Float;
20170: Function SinH( const Z : Float) : Float;
20171: Function TanH( const Z : Float) : Float;
20172:
20173: //*****from DMath.Dll Lib of types.inc in source\dmath_dll
20174: InvLn2           = 1.44269504088896340736; { 1/Ln(2) }
20175: InvLn10          = 0.43429448190325182765; { 1/Ln(10) }
20176: TwoPi           = 6.28318530717958647693; { 2*Pi }
20177: PiDiv2          = 1.57079632679489661923; { Pi/2 }
20178: SqrtPi          = 1.77245385090551602730; { Sqrt(Pi) }
20179: Sqrt2Pi          = 2.50662827463100050242; { Sqrt(2*Pi) }
20180: InvSqrt2Pi       = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
20181: LnSqrt2Pi        = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
20182: Ln2PiDiv2        = 0.91893853320467274178; { Ln(2*Pi)/2 }
20183: Sqrt2             = 1.41421356237309504880; { Sqrt(2) }
20184: Sqrt2Div2        = 0.70710678118654752440; { Sqrt(2)/2 }
20185: Gold             = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
20186: CGold            = 0.38196601125010515179; { 2 - GOLD }
20187: MachEp           = 2.220446049250313E-16; { 2^(-52) }
20188: MaxNum           = 1.797693134862315E+308; { 2^1024 }
20189: MinNum           = 2.225073858507202E-308; { 2^(-1022) }
20190: MaxLog           = 709.7827128933840;
20191: MinLog           = -708.3964185322641;
20192: MaxFac           = 170;
20193: MaxGam           = 171.624376956302;
20194: MaxLgm           = 2.556348E+305;
20195: SingleCompareDelta = 1.0E-34;
20196: DoubleCompareDelta = 1.0E-280;
20197: {$IFDEF CLR}
20198: ExtendedCompareDelta = DoubleCompareDelta;
20199: {$ELSE}
20200: ExtendedCompareDelta = 1.0E-4400;
20201: {$ENDIF}
20202: Bytes1KB          = 1024;
20203: Bytes1MB          = 1024 * Bytes1KB;
20204: Bytes1GB          = 1024 * Bytes1MB;
20205: Bytes64KB         = 64 * Bytes1KB;
20206: Bytes64MB         = 64 * Bytes1MB;
20207: Bytes2GB          = 2 * LongWord(Bytes1GB);
20208: clBlack32' , $FF000000 );
20209: clDimGray32' , $FF3F3F3F );
20210: clGray32' , $FF7F7F7F );
20211: clLightGray32' , $FFBFBFBF );
20212: clWhite32' , $FFFFFF );
20213: clMaroon32' , $FF7F0000 );
20214: clGreen32' , $FF007F00 );
20215: clOlive32' , $FF7F7F00 );
20216: clNavy32' , $FF00007F );
20217: clPurple32' , $FF7F007F );
20218: clTeal32' , $FF007F7F );
20219: clRed32' , $FFFF0000 );
20220: clLime32' , $FF00FF00 );
20221: clYellow32' , $FFFFFF00 );
20222: clBlue32' , $FF0000FF );
20223: clFuchsia32' , $FFFF00FF );
20224: clAqua32' , $FF00FFFF );
20225: clAliceBlue32' , $FFFF0F8F );
20226: clAntiqueWhite32' , $FFFAEBD7 );
20227: clAquamarine32' , $FF7FFF4 );
20228: clAzure32' , $FFF0FFFF );
20229: clBeige32' , $FFF5F5DC );
20230: clBisque32' , $FFFE4C4 );
20231: clBlanchedAlmond32' , $FFFFEBBC );
20232: clBlueViolet32' , $FF8A2BE2 );
20233: clBrown32' , $FFA52A2A );
20234: clBurlyWood32' , $FFDEB887 );
20235: clCadetblue32' , $FF5F9EA0 );
20236: clChartreuse32' , $FF7FFF00 );
20237: clChocolate32' , $FFD2691E );
20238: clCoral32' , $FFFF7F50 );
20239: clCornFlowerBlue32' , $FF6495ED );
20240: clCornSilk32' , $FFFFF8DC );
20241: clCrimson32' , $FFDC143C );
20242: clDarkBlue32' , $FF00008B );
20243: clDarkCyan32' , $FF008B8B );
20244: clDarkGoldenRod32' , $FFB8860B );
20245: clDarkGray32' , $FFA9A9A9 );
20246: clDarkGreen32' , $FF006400 );

```

```
20247:    clDarkGrey32', $FFA9A9A9 ));  
20248:    clDarkKhaki32', $FFbdb76B ));  
20249:    clDarkMagenta32', $FF8B008B ));  
20250:    clDarkOliveGreen32', $FF556B2F ));  
20251:    clDarkOrange32', $FFFF8C00 ));  
20252:    clDarkOrchid32', $FF9932CC ));  
20253:    clDarkRed32', $FF8B0000 ));  
20254:    clDarkSalmon32', $FFE9967A ));  
20255:    clDarkSeaGreen32', $FF8FBBC8F ));  
20256:    clDarkSlateBlue32', $FF483D8B ));  
20257:    clDarkSlateGray32', $FF2F4F4F ));  
20258:    clDarkSlateGrey32', $FF2F4F4F ));  
20259:    clDarkTurquoise32', $FF00CED1 ));  
20260:    clDarkViolet32', $FF9400D3 ));  
20261:    clDeepPink32', $FFFF1493 ));  
20262:    clDeepSkyBlue32', $FF00BFFF ));  
20263:    clDodgerBlue32', $FF1E90FF ));  
20264:    clFireBrick32', $FFB22222 ));  
20265:    clFloralWhite32', $FFFFFFAF0 ));  
20266:    clGainsboro32', $FFDCDCDC ));  
20267:    clGhostWhite32', $FFF8F8FF ));  
20268:    clGold32', $FFFFFD700 ));  
20269:    clGoldenRod32', $FFDA5A50 ));  
20270:    clGreenYellow32', $FFADFF2F ));  
20271:    clGrey32', $FF808080 ));  
20272:    clHoneyDew32', $FFF0FFF0 ));  
20273:    clHotPink32', $FFFF69B4 ));  
20274:    clIndianRed32', $FFCD5C5C ));  
20275:    clIndigo32', $FF4B0082 ));  
20276:    clIvory32', $FFFFFFF0 ));  
20277:    clKhaki32', $FFF0E68C ));  
20278:    clLavender32', $FFE6E6FA ));  
20279:    clLavenderBlush32', $FFFFFF0F5 ));  
20280:    clLawnGreen32', $FF7CFC00 ));  
20281:    clLemonChiffon32', $FFFFFFACD ));  
20282:    clLightBlue32', $FFADD8E6 ));  
20283:    clLightCoral32', $FFF08080 ));  
20284:    clLightCyan32', $FFEOF0FFF ));  
20285:    clLightGoldenRodYellow32', $FFFAFAD2 ));  
20286:    clLightGreen32', $FF90EE90 ));  
20287:    clLightGrey32', $FFD3D3D3 ));  
20288:    clLightPink32', $FFFFB6C1 ));  
20289:    clLightSalmon32', $FFFA07A ));  
20290:    clLightSeagreen32', $FF20B2AA ));  
20291:    clLightSkyblue32', $FF87CEFA ));  
20292:    clLightSlategray32', $FF778899 ));  
20293:    clLightSlategrey32', $FF778899 ));  
20294:    clLightSteelblue32', $FFB0C4DE ));  
20295:    clLightYellow32', $FFFFFFE0 ));  
20296:    clLtGray32', $FFC0COC0 ));  
20297:    clMedGray32', $FFAO0A0A4 ));  
20298:    clDkGray32', $FF808080 ));  
20299:    clMoneyGreen32', $FFC0DCC0 ));  
20300:    clLegacySkyBlue32', $FFA6CAF0 ));  
20301:    clCream32', $FFFFFFBF0 ));  
20302:    clLimeGreen32', $FF32CD32 ));  
20303:    clLinen32', $FFFAF0E6 ));  
20304:    clMediumAquamarine32', $FFP66CDAA ));  
20305:    clMediumBlue32', $FF0000CD ));  
20306:    clMediumOrchid32', $FFBA55D3 ));  
20307:    clMediumPurple32', $FF9370DB ));  
20308:    clMediumSeaGreen32', $FF3CB371 ));  
20309:    clMediumSlateBlue32', $FF7B68EE ));  
20310:    clMediumSpringGreen32', $FF00FA9A ));  
20311:    clMediumTurquoise32', $FF48D1CC ));  
20312:    clMediumVioletRed32', $FFC71585 ));  
20313:    clMidnightBlue32', $FF191970 ));  
20314:    clMintCream32', $FFFF5FFA ));  
20315:    clMistyRose32', $FFFFE4E1 ));  
20316:    clMoccasin32', $FFFFE4B5 ));  
20317:    clNavajoWhite32', $FFFDFEAD ));  
20318:    clOldLace32', $FFDDF5E6 ));  
20319:    clOliveDrab32', $FF6B8E23 ));  
20320:    clOrange32', $FFFA500 ));  
20321:    clOrangeRed32', $FFFF4500 ));  
20322:    clOrchid32', $FFDA70D6 ));  
20323:    clPaleGoldenRod32', $FFEEE8AA ));  
20324:    clPaleGreen32', $FF98FB98 ));  
20325:    clPaleTurquoise32', $FFAFEEEE ));  
20326:    clPaleVioletred32', $FFDB7093 ));  
20327:    clPapayaWhip32', $FFFFEFD5 ));  
20328:    clPeachPuff32', $FFFFDAB9 ));  
20329:    clPeru32', $FFCD853F ));  
20330:    clPlum32', $FFDAA0DD ));  
20331:    clPowderBlue32', $FFB0E0E6 ));  
20332:    clRosyBrown32', $FFB8F8F ));  
20333:    clRoyalBlue32', $FF4169E1 ));  
20334:    clSaddleBrown32', $FF8B4513 ));  
20335:    clSalmon32', $FFFA8072 ));
```

```

20336:   clSandyBrown32', $FFF4A460 ));
20337:   clSeaGreen32', $FF2E8B57 ));
20338:   clSeaShell32', $FFFFF5EE ));
20339:   clSienna32', $FFA0522D ));
20340:   clSilver32', $FFC0C0C0 ));
20341:   clSkyblue32', $FF87CEEB ));
20342:   clSlateBlue32', $FF6A5ACD ));
20343:   clSlateGray32', $FF708090 ));
20344:   clSlateGrey32', $FF708090 ));
20345:   clSnow32', $FFFFFFFAFA ));
20346:   clSpringgreen32', $FF00FF7F ));
20347:   clSteelblue32', $FF4682B4 ));
20348:   clTan32', $FD2B48C ));
20349:   clThistle32', $FFD8BFD8 ));
20350:   clTomato32', $FFFF6347 ));
20351:   clTurquoise32', $FF40E0D0 ));
20352:   clViolet32', $FFEE82EE ));
20353:   clWheat32', $FFF5DEB3 ));
20354:   clWhitesmoke32', $FFF5F5F5 ));
20355:   clYellowgreen32', $FF9ACD32 ));
20356:   clTrWhite32', $7FFFFFFF ));
20357:   clTrBlack32', $F00000000 ));
20358:   clTrRed32', $7FFF00000000 ));
20359:   clTrGreen32', $7FO0FF00000000 ));
20360:   clTrBlue32', $7FO00000F00000000 ));
20361: // Fixed point math constants
20362: FixedOne = $10000; FixedHalf = $7FFF;
20363: FixedPI = Round(PI * FixedOne);
20364: FixedToFloat = 1/FixedOne;
20365:
20366: Special Types
20367: ****
20368: type Complex = record //for complex numbers
20369:   X, Y : Float;
20370: end;
20371: type TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R : '
20372:   + Float; Theta : Float; end');
20373: type TVector' = array of Float;
20374: TIntVector' = array of Integer;
20375: TCompVector' = array of Complex;
20376: TBoolVector' = array of Boolean;
20377: TStringVector' = array of String;
20378: TMatrix' = array of TVector';
20379: TIntMatrix' = array of TIntVector';
20380: TCompMatrix' = array of TCompVector';
20381: TBoolMatrix' = array of TBoolVector';
20382: TStringMatrix' = array of TStringVector';
20383: TByteArray' = array[0..32767] of byte; !
20384: THexArray' = array [0..15] of Char; // = '0123456789ABCDEF';
20385: TBitmapStyle = (bsNormal, bsCentered, bsStretched);
20386: T2StringArray' = array of array of string;
20387: T2IntegerArray' = array of array of integer;
20388: AddTypeS('INT_PTR', 'Integer
20389: AddTypeS('LONG_PTR', 'Integer
20390: AddTypeS('UINT_PTR', 'Cardinal
20391: AddTypeS('ULONG_PTR', 'Cardinal
20392: AddTypeS('DWORD_PTR', 'ULONG_PTR
20393: TIntegeDynArray', 'array of Integer
20394: TCardinalDynArray', 'array of Cardinal
20395: TWordDynArray', 'array of Word
20396: TSmallIntDynArray', 'array of SmallInt
20397: TByteDynArray', 'array of Byte
20398: TShortIntDynArray', 'array of ShortInt
20399: TInt64DynArray', 'array of Int64
20400: TLongWordDynArray', 'array of LongWord
20401: TSinglDynArray', 'array of Single
20402: TDoubleDynArray', 'array of Double
20403: TBooleanDynArray', 'array of Boolean
20404: TStringDynArray', 'array of string
20405: TWideStringDynArray', 'array of WideString
20406: TDynByteArray' = array of Byte;
20407: TDynShortintArray' = array of Shortint;
20408: TDynSmallintArray' = array of Smallint;
20409: TDynWordArray' = array of Word;
20410: TDynIntegerArray' = array of Integer;
20411: TDynLongintArray' = array of Longint;
20412: TDynCardinalArray' = array of Cardinal;
20413: TDynInt64Array' = array of Int64;
20414: TDynExtendedArray' = array of Extended;
20415: TDynDoubleArray' = array of Double;
20416: TDynSingleArray' = array of Single;
20417: TDynFloatArray' = array of Float;
20418: TDynPointerArray' = array of Pointer;
20419: TDynStringArray' = array of string;
20420: TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
20421:   ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
20422: TSynSearchOptions = set of TSynSearchOption;
20423: TFloat = single
20424: Float = double

```

```

20425:
20426:
20427: /* Project: IFSI_WinFormIpuzzle.pas BaseInclude RunTimeLib for maxbox *: pas_includebox.inc
20428: -----
20429: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
20430: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
20431: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
20432: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
20433: function CheckStringSum(vstring: string): integer;
20434: function HexToInt(HexNum: string): LongInt;
20435: function IntToBin(Int: Integer): String;
20436: function BinToInt(Binary: String): Integer;
20437: function HexToBin(HexNum: string): string; external2
20438: function BinToHex(Binary: String): string;
20439: function IntToFloat(i: Integer): double;
20440: function AddThousandSeparator(S: string; myChr: Char): string;
20441: function Max3(const X,Y,Z: Integer): Integer;
20442: procedure Swap(var X,Y: char); // faster without inline
20443: procedure ReverseString(var S: String);
20444: function CharToHexStr(Value: Char): string;
20445: function CharToUniCode(Value: Char): string;
20446: function Hex2Dec(Value: Str002): Byte;
20447: function HexStrCodeToStr(Value: string): string;
20448: function HexToStr(i: integer; value: string): string;
20449: function UniCodeToStr(Value: string): string;
20450: function CRC16(statement: string): string;
20451: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
20452: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
20453: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
20454: Procedure ExecuteCommand(executeFile, paramstring: string);
20455: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
20456: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
20457: procedure SearchAndOpenDoc(vfilenamepath: string);
20458: procedure ShowInterfaces(myFile: string);
20459: function Fact2(av: integer): extended;
20460: Function BoolToStr(B: Boolean): string;
20461: Function GCD(x, y : LongInt) : LongInt;
20462: function LCM(m,n: longint): longint;
20463: function GetASCII: string;
20464: function GetItemHeight(Font: TFont): Integer;
20465: function myPlaySound(s: pchar; flag, syncflag: integer): boolean;
20466: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
20467: function getHINSTANCE: longword;
20468: function getHMODULE: longword;
20469: function GetASCII: string;
20470: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
20471: function WordIsOk(const AWord: string; var VW: Word): boolean;
20472: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
20473: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
20474: function SafeStr(const s: string): string;
20475: function ExtractUrlPath(const FileName: string): string;
20476: function ExtractUrlName(const FileName: string): string;
20477: function IsInternet: boolean;
20478: function RotateLeft1Bit_u32( Value: uint32): uint32;
20479: procedure LinearRegression(const KnownY:array of Double;const KnownX:array of Double;NData:Int;var LF:TStLinEst; ErrorStats: Bool);
20480: procedure getEnvironmentInfo;
20481: procedure AntiFreeze;
20482: function GetCPUSpeed: Double;
20483: function IsVirtualPcGuest : Boolean;
20484: function IsVmWareGuest : Boolean;
20485: procedure StartSerialDialog;
20486: function IsWoW64: boolean;
20487: function IsWow64String(var s: string): Boolean;
20488: procedure StartThreadDemo;
20489: Function RGB(R,G,B: Byte): TColor;
20490: Function Sendln(amess: string): boolean;
20491: Procedure maxbox;
20492: Function AspectRatio(aWidth, aHeight: Integer): String;
20493: function wget(aURL, afile: string): boolean;
20494: procedure PrintList(Value: TStringList);
20495: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
20496: procedure getEnvironmentInfo;
20497: procedure AntiFreeze;
20498: function getBitmap(apath: string): TBitmap;
20499: procedure ShowMessageBig(const aText : string);
20500: function YesNoDialog(const ACaption, AMsg: string): boolean;
20501: procedure SetArrayLength2String(arr: T2StringArray; asizel, asize2: integer);
20502: procedure SetArrayLength2Integer(arr: T2IntegerArray; asizel, asize2: integer);
20503: //function myStrToBytes(const Value: String): TBytes;
20504: //function myBytesToStr(const Value: TBytes): String;
20505: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20506: function getBitmap(apath: string): TBitmap;
20507: procedure ShowMessageBig(const aText : string);
20508: Function StrToBytes(const Value: String): TBytes;
20509: Function BytesToStr(const Value: TBytes): String;
20510: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20511: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retries:Int;var HostName:String):Bool;

```

```

20512: function FindInPaths(const fileName, paths : String) : String;
20513: procedure initHexArray(var hexn: THexArray);
20514: function josephusG(n,k: integer; var graphout: string): integer;
20515: function isPowerof2(num: int64): boolean;
20516: function powerOf2(exponent: integer): int64;
20517: function getBigPI: string;
20518: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
20519: function GetASCIIline: string;
20520: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
20521:                                pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
20522: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
20523: procedure AddComplexSoundObjectToList(newf,newp,newa,neww:integer; freqlist: TStrings);
20524: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
20525: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
20526: function isKeypressed: boolean;
20527: function Keypress: boolean;
20528: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
20529: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
20530: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
20531: function GetOSName: string;
20532: function GetOSVersion: string;
20533: function GetOSNumber: string;
20534: function getEnvironmentString: string;
20535: procedure StrReplace(var Str: String; Old, New: String);
20536: procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
20537: function getTeamViewerID: string;
20538: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
20539: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
20540: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
20541: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
20542: function StartSocketService: Boolean;
20543: procedure StartSocketServiceForm;
20544: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
20545: function GetFileList1(apath: string): TStringlist;
20546: procedure LetfileList(FileList: TStringlist; apath: string);
20547: procedure StartWeb(aurl: string);
20548: function GetTodayFiles(startdir, amask: string): TStringlist;
20549: function PortTCPISOpen(dwPort : Word; ipAddressStr: String): boolean;
20550: function JavahashCode(val: string): Integer;
20551: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
20552: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
20553: Procedure HideWindowForSeconds(secs: integer); //3 seconds
20554: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); //3 seconds
20555: Procedure ConvertToGray(Cnv: TCanvas);
20556: function GetFileDate(aFile:string; aWithTime:Boolean):string;
20557: procedure ShowMemory;
20558: function ShowMemory2: string;
20559: function getHostIP: string;
20560: procedure ShowBitmap(bmap: TBitmap);
20561: function GetOsVersionInfo: TOSVersionInfo; //thx to wischnewski
20562: function CreateDBGridForm(dblist: TStringList): TListBox;
20563: function isService: boolean;
20564: function isApplication: boolean;
20565: function isTerminalSession: boolean;
20566: function SetPrivilege(privilegeName: string; enable: boolean): boolean;
20567: procedure getScriptandRunAsk;
20568: procedure getScriptandRun(ascript: string);
20569: function VersionCheckAct: string;
20570: procedure getBox(aurl, extension: string);
20571: function CheckBox: string;
20572: function isNTFS: boolean;
20573: //procedure doWebCamPic;
20574: procedure doWebCamPic(picname: string);
20575: function readm: string;
20576: procedure getGEOMapandRunAsk;
20577: function GetMapX(C_form,apath: string; const Data: string): boolean;
20578: procedure GetGEOMap(C_form,apath: string; const Data: string);
20579: function GetMapXGeoReverse(C_form: string; const lat,long: string): string;
20580: //function RoundTo(const AValue: Extended;
20581: //                           const ADigit: TRoundToEXRangeExtended): Extended;
20582: function Downloadfile(SourceFile, DestFile: string): Boolean;
20583: function DownloadFileOpen(SourceFile, DestFile: string): Boolean;
20584: function OpenMap(const Data: string): boolean;
20585: function GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
20586: Function getFileCount(amask: string): integer;
20587: function CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TNavPos): string;
20588: procedure Debugln(DebugLogFile: string; E: string);
20589: function IntToFloat(i: Integer): double;
20590: function AddThousandSeparator(S: string; myChr: Char): string;
20591: function mymcisendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
20592: function RoundTime(ADate: string; Rounding: Integer; bRound: Boolean): string;
20593: End C:\maxbook\maxbox3\mX3999\maxbox3\source\IFSI_WinForm1puzzle.pas File loaded
20594:
20595: // News of 3.9.8 up
20596: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20597: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20598: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20599: JvChart - TJvChart Component - 2009 Public
20600: MemoryLeakReport in ini-file (MEMORYREPORT=Y)

```

```

20601: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
20602: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
20603: DMath DLL included incl. Demos
20604: Interface Navigator menu/View/Intf Navigator
20605: Unit Explorer menu/Debug/Units Explorer
20606: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maxBox
20607: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
20608: Script History to 9 Files WebServer light /Options/Addons/WebServer
20609: Full Text Finder, JVSSimLogic Simulator Package
20610: Halt-Stop Program in Menu, WebServer2, Stop Event ,
20611: Conversion Routines, Prebuild Forms, CodeSearch
20612: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20613: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20614: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20615: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLLoader, TJvPaintFX
20616: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
20617: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
20618: IDE Reflection API, Session Service Shell S3
20619: additional SynEdit API, IsKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
20620: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
20621: arduino map() function, PRMRandom Generator
20622: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
20623: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
20624: REST Test Lib, Multilang Component, Forth Interpreter
20625: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
20626: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
20627: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20628: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20629: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
20630: QRCode Service, add more CFunctions like CDateTime of Synapse
20631: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
20632: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
20633: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
20634: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
20635: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
20636: BOLD Package, Indy Package5, maTRIX. MATHMAX
20637: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
20638: emax layers: system-package-component-unit-class-function-block
20639: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms
20640: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
20641: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
20642: OpenGL Game Demo: ..Options/Add Ons/Reversi
20643: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
20644: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
20645: 7% performance gain (hot spot profiling)
20646: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
20647: add 42 + 22 (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
20648: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
20649: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
20650:
20651: add routines in 3.9.7.5
20652: 097: procedure RIRegister_BarCodeScanner_Routines(S: TPSEExec);
20653: 996: procedure RIRegister_DBCtrls_Routines(S: TPSEExec);
20654: 069: procedure RIRegister_IdStrings_Routines(S: TPSEExec);
20655: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSEExec);
20656: 215: procedure RIRegister_PNGLoader_Routines(S: TPSEExec);
20657: 374: procedure RIRegister_SerDlgs_Routines(S: TPSEExec);
20658: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSEExec);
20659:
20660: ////////////////////////////// TestUnits //////////////////////////////
20661: SelftestPEM;
20662: SelfTestCFundamentUtils;
20663: SelfTestCFileUtils;
20664: SelfTestCDatetime;
20665: SelfTestCTimer;
20666: SelfTestCRandom;
20667: Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter'
20668:           Assert(WinPathToUnixPath('\c\d\f') = '/c/d/f', 'WinPathToUnixPath'
20669:
20670: // Note: There's no need for installing a client certificate in the
20671: // webbrowser. The server asks the webbrowser to send a certificate but
20672: // if nothing is installed the software will work because the server
20673: // doesn't check to see if a client certificate was supplied. If you want you can install:
20674: // file: c_cacert.p12 password: c_cakey
20675:
20676: TGraphicControl = class(TControl)
20677: private
20678:   FCanvas: TCanvas;
20679:   procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20680: protected
20681:   procedure Paint; virtual;
20682:   property Canvas: TCanvas read FCanvas;
20683: public
20684:   constructor Create(AOwner: TComponent); override;
20685:   destructor Destroy; override;
20686: end;
20687:
20688: TCustomControl = class(TWinControl)
20689: private

```

```

20690:     FCanvas: TCanvas;
20691:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20692:   protected
20693:     procedure Paint; virtual;
20694:     procedure PaintWindow(DC: HDC); override;
20695:     property Canvas: TCanvas read FCanvas;
20696:   public
20697:     constructor Create(AOwner: TComponent); override;
20698:     destructor Destroy; override;
20699:   end;
20700:   RegisterPublishedProperties;
20701:   ('ONCHANGE', 'TNotifyEvent', iptrw);
20702:   ('ONCLICK', 'TNotifyEvent', iptrw);
20703:   ('ONDBLCLICK', 'TNotifyEvent', iptrw);
20704:   ('ONENTER', 'TNotifyEvent', iptrw);
20705:   ('ONEXIT', 'TNotifyEvent', iptrw);
20706:   ('ONKEYDOWN', 'TKeyEvent', iptrw);
20707:   ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
20708:   ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
20709:   ('ONMOUSEMOVE', 'TMouseEvent', iptrw);
20710:   ('ONMOUSEUP', 'TMouseEvent', iptrw);
20711: //*****
20712: // To stop the while loop, click on Options>Show Include (boolean switch) !
20713: Control a loop in a script with a form event:
20714: IncludeON; //control the while loop
20715: while maxform1.ShowInclude1.checked do begin //menu event Options>Show Include
20716: repeat {for it:= 1 to n do} until is keypressed //keypress in output window below (memo2)
20717:
20718: //-----
20719: //*****mX4 ini-file Configuration*****
20720: //-----
20721: using config file maxboxdef.ini      menu/Help/Config File
20722:
20723: //*** Definitions for maXbox mX3 ***
20724: [FORM]
20725: LAST_FILE=E:\maxbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
20726: FONTSIZE=14
20727: EXTENSION=txt
20728: SCREENX=1386
20729: SCREENY=1077
20730: MEMHEIGHT=350
20731: PRINTFONT=Courier New //GUI Settings
20732: LINENUMBERS=Y //line numbers at gutter in editor at left side
20733: EXCEPTIONLOG=Y //store excepts + success in 2 log files see below! -menu Debug/Show Last Exceptions
20734: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
20735: BOOTSCRIPT=Y //enabling load a boot script
20736: MEMORYREPORT=Y //shows memory report on closing maXbox
20737: MACRO=Y //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
20738: NAVIGATOR=N //shows function list at the right side of editor
20739: NAVWIDTH=350 //width of the right side interface list <CTRL L> >=200
20740: AUTOBOOKMARK=Y //sets on all functions a bookmark to jump
20741: [WEB]
20742: IPPORT=8080 //for internal webserver - menu /Options/Add Ons/WebServer
20743: IPHOST=192.168.1.53 //run as Administrator!
20744: ROOTCERT=filepathY
20745: SCERT=filepathY
20746: RSAKEY=filepathY
20747: VERSIONCHECK=Y
20748: APP=C:\WINDOWS\System32\calc.exe //set path to an external app
20749:
20750:
20751: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
20752: Also possible to set report memory in script to override ini setting
20753: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
20754:
20755: After Change the ini file you can reload the file with ..../Help/Config Update
20756:
20757: //-----
20758: //*****mX4 maildef.ini ini-file Configuration*****
20759: //-----
20760: //*** Definitions for maXMail ***
20761: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
20762: [MAXMAIL]
20763: HOST=getmail.softwareschule.ch
20764: USER=mailusername
20765: PASS=password
20766: PORT=110
20767: SSL=Y
20768: BODY=Y
20769: LAST=5
20770:
20771: ADO Connection String:
20772: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
20773: \452_dbtreeview2access.txt \452_dbtrv3accessUML2.txt
20774: program TestDbTreeViewMainForm2_ACCESS;
20775: ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
20776:           +Exepath+'\examples\detail.mdb;Persist Security Info=False';
20777:           'Provider=MSDASQL.1;Persist Security Info=False:Extended
Properties="DSN=FB_EMPLOYEE;Driver=Firebird/InterBase(r)
driver;Dbname=C:\maxbook\maxbox3\mX3999\maxbox3\examples\EMPLOYEE.FDB;CHARSET=NONE;UID=SYSDBA;Role=Admin";'

```

```

20778:
20779: OpenSSL Lib: unit ssl_openssl_lib;
20780: {$IFDEF CIL}
20781: const
20782: {$IFDEF LINUX}
20783: DLLSSLName = 'libssl.so';
20784: DLLUtilName = 'libcrypto.so';
20785: {$ELSE}
20786: DLLSSLName = 'ssleay32.dll';
20787: DLLUtilName = 'libleay32.dll';
20788: {$ENDIF}
20789: {$ELSE}
20790: var
20791: {$IFNDEF MSWINDOWS}
20792: {$IFDEF DARWIN}
20793: DLLSSLName: string = 'libssl.dylib';
20794: DLLUtilName: string = 'libcrypto.dylib';
20795: {$ELSE}
20796: DLLSSLName: string = 'libssl.so';
20797: DLLUtilName: string = 'libcrypto.so';
20798: {$ENDIF}
20799: {$ELSE}
20800: DLLSSLName: string = 'ssleay32.dll';
20801: DLLSSLName2: string = 'libssl32.dll';
20802: DLLUtilName: string = 'libleay32.dll';
20803: {$ENDIF}
20804: {$ENDIF}
20805:
20806: //-----
20807: //*****mX4 Macro Tags *****
20808: //-----
20809:
20810: asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
E:\maxbox\maxbox3\docs\maxbox_extract_funcList399.txt end
20811:
20812: //Tag Macros in ini-file configure
20813:
20814: asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
20815:
20816: //Tag Macros
20817: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
20818: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
20819: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
20820: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
20821: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
20822: 10193: SearchAndCopy(memo1.lines, '#files', fname + '+' + SHA1(Act_Filename), 11);
20823: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
20824: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
20825: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
20826: [getUserNameWin, getComputernameWin, datetimetoStr(now),
20827: 10196: SearchAndCopy(memo1.lines, '#head', Format('%s: %s: %s %s',
20828: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]], 11));
20829: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]], 11));
20830: 10198: SearchAndCopy(memo1.lines, '#tech', Format('perf: %s threads: %d %s %s',
20831: [perftime, numprocesssthreads, getAddress(getComputerNameWin), timetoStr(time), mbversion]], 11));
20832: 10298: SearchAndCopy(memo1.lines, '#net', Format('DNS: %s; local IPs: %s; local IP: %s',
20833: [getDNS, GetLocalIPs, getAddress(getComputerNameWin)]], 10));
20834:
20835: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
20836:
20837: //Replace Macros
20838: SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
20839: SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
20840: SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
20841: SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPATH, 9);
20842: SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
20843: SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
20844: ref: netcologne.dl.sourceforge.net/project/maxbox/maxbox3.zip
20845: SearchAndCopy(memo1.lines, '#tech'perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
20846: [perftime,numprocesssthreads,address(getComputerNameWin),timetoStr(time),mbversion]], 11);
20847: //#tech!84perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
20848: SearchAndCopy(memo1.lines, 'maxbox_extract_funcList399.txt'
20849:
20850: //-----
20851: //*****mX4 ToDo List Tags ..../Help/ToDo List*****
20852: //-----
20853:
20854: while I < sl.Count do begin
20855: // if MatchesMask(sl[I], '/? TODO ([a-zA-Z_]*#[1-9#]*:*)') then
20856: if MatchesMask(sl[I], '/? TODO (?*#?#)*:*)' ) then
20857: BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
20858: else if MatchesMask(sl[I], '/? DONE (?*#?#)*:*)' ) then
20859: BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
20860: else if MatchesMask(sl[I], '/? TODO (#?#)*:*)' ) then
20861: BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
20862: else if MatchesMask(sl[I], '/? DONE (#?#)*:*)' ) then
20863: BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
20864: else if MatchesMask(sl[I], '/?*?TODO*:*)' ) then
20865: BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO

```

```

20866:     else if MatchesMask(sl[I], '*/?*DONE*:*)' then
20867:       BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
20868:       Inc(I);
20869:     end;
20870:
20871: //-----
20872: //*****mX4 Public Tools API *****
20873: //-----
20874:   file : unit uPSI_fMain.pas;           {$OTAP} Open Tools API Catalog
20875: // Those functions concern the editor and preprocessor, all of the IDE
20876: Example: Call it with maxform1.InfoClick(self)
20877: Note: Call all Methods with maxForm1., e.g.:
20878:           maxForm1.ShellStyle1Click(self);
20879:
20880: procedure SIRegister_fMain(CL: TPSPascalCompiler);
20881: begin
20882:   Const('BYTECODE','String bytecode.txt'
20883:   Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT')
20884:   Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC'
20885:   Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS'
20886:   Const('PSINC','String PS Includes (*.inc)|*.INC'
20887:   Const('DEFFILENAME','String 'firstdemo.txt'
20888:   Const('DEFINIFILE','String 'maxboxdef.ini'
20889:   Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt'
20890:   Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt'
20891:   Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf'
20892:   Const('ALLOBJECTSLIST','String 'docs\VCL.pdf'
20893:   Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt'
20894:   Const('ALLUNITLIST','String 'docs\maxbox3_9.xml')
20895:   Const('INCLUDEBOX','String 'pas_includebox.inc'
20896:   Const('BOOTSCRIPT','String 'maxbootscript.txt'
20897:   Const('MBVERSION','String '3.9.9.190'
20898:   Const('VERSION','String '3.9.9.190'
20899:   Const('MBVER','String '399'
20900:   Const('MBVER1','Integer'(399);
20901:   Const('MBVERIALL','Integer'(399190);
20902:   Const('EXENAME','String 'maxBox3.exe'
20903:   Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm'
20904:   Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt'
20905:   Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt'
20906:   Const('MXINTERNETCHECK','String 'www.ask.com'
20907:   Const('MXMAIL','String 'max@kleiner.com'
20908:   Const('TAB','Char #$09)';
20909:   Const('CODECOMPLETION','String 'bds_delphi.dci'
20910:   SIRegister_TMaxForm1(CL);
20911: end;
20912:
20913: with FindClass('TForm'),'TMaxForm1') do begin
20914:   memo2, 'TMemo', iptrw;
20915:   memo1, 'TSynMemo', iptrw;
20916:   CB1SCList, 'TComboBox', iptrw);
20917:   mxNavigator, 'TComboBox', iptrw);
20918:   IPHost, 'string', iptrw);
20919:   IPPort, 'integer', iptrw);
20920:   COMPort, 'integer', iptrw); //3.9.6.4
20921:   Splitter1, 'TSplitter', iptrw);
20922:   PSScript, 'TPSScript', iptrw);
20923:   PS3DllPlugin, 'TPSDllPlugin', iptrw);
20924:   MainMen1, 'TMainMenu', iptrw);
20925:   Program1, 'TMenuItem', iptrw);
20926:   Compile1, 'TMenuItem', iptrw);
20927:   Files1, 'TMenuItem', iptrw);
20928:   open1, 'TMenuItem', iptrw);
20929:   Save2, 'TMenuItem', iptrw);
20930:   Options1, 'TMenuItem', iptrw);
20931:   Savebefore1, 'TMenuItem', iptrw);
20932:   Largefont1, 'TMenuItem', iptrw);
20933:   sBytecode1, 'TMenuItem', iptrw);
20934:   Saveas3, 'TMenuItem', iptrw);
20935:   Clear1, 'TMenuItem', iptrw);
20936:   Slinenumbers1, 'TMenuItem', iptrw);
20937:   About1, 'TMenuItem', iptrw);
20938:   Search1, 'TMenuItem', iptrw);
20939:   SynPasSyn1, 'TSynPasSyn', iptrw);
20940:   memo1, 'TSynMemo', iptrw);
20941:   SynEditSearch1, 'TSynEditSearch', iptrw);
20942:   WordWrap1, 'TMenuItem', iptrw);
20943:   XPMManifest1, 'TXPManifest', iptrw);
20944:   SearchNext1, 'TMenuItem', iptrw);
20945:   Replace1, 'TMenuItem', iptrw);
20946:   PSImport_Controls1, 'TPSImport_Controls', iptrw);
20947:   PSImport_Classes1, 'TPSImport_Classes', iptrw);
20948:   ShowInclude1, 'TMenuItem', iptrw);
20949:   SynEditPrint1, 'TSynEditPrint', iptrw);
20950:   Printout1, 'TMenuItem', iptrw);
20951:   mnPrintColors1, 'TMenuItem', iptrw);
20952:   dlgFilePrint1, 'TPrintDialog', iptrw);
20953:   dlgPrintFont1, 'TFontDialog', iptrw);
20954:   mnuPrintFont1, 'TMenuItem', iptrw);

```

```
20955:     Include1', 'TMenuItem', iptrw);
20956:     CodeCompletionList1', 'TMenuItem', iptrw);
20957:     IncludeList1', 'TMenuItem', iptrw);
20958:     ImageList1', 'TImageList', iptrw);
20959:     ImageList2', 'TImageList', iptrw);
20960:     CoolBar1', 'TCoolBar', iptrw);
20961:     ToolBar1', 'TToolBar', iptrw);
20962:     btnLoad', 'TToolButton', iptrw);
20963:     ToolButton2', 'TToolButton', iptrw);
20964:     btnFind', 'TToolButton', iptrw);
20965:     btnCompile', 'TToolButton', iptrw);
20966:     btnTrans', 'TToolButton', iptrw);
20967:     btnUseCase', 'TToolButton', iptrw); //3.8
20968:     toolbtnTutorial', 'TToolButton', iptrw);
20969:     btn6res', 'TToolButton', iptrw);
20970:     ToolButton5', 'TToolButton', iptrw);
20971:     ToolButton1', 'TToolButton', iptrw);
20972:     ToolButton3', 'TToolButton', iptrw);
20973:     statusBar1', 'TStatusBar', iptrw);
20974:     SaveOutput1', 'TMenuItem', iptrw);
20975:     ExportClipboard1', 'TMenuItem', iptrw);
20976:     Close1', 'TMenuItem', iptrw);
20977:     Manual1', 'TMenuItem', iptrw);
20978:     About2', 'TMenuItem', iptrw);
20979:     loadLastfile1', 'TMenuItem', iptrw);
20980:     imgLogo', 'TImage', iptrw);
20981:     cedebug', 'TPSScriptDebugger', iptrw);
20982:     debugPopupMenul', 'TPopupMenu', iptrw);
20983:     BreakPointMenu1', 'TMenuItem', iptrw);
20984:     Decompile1', 'TMenuItem', iptrw);
20985:     StepInto1', 'TMenuItem', iptrw);
20986:     StepOut1', 'TMenuItem', iptrw);
20987:     Reset1', 'TMenuItem', iptrw);
20988:     DebugRun1', 'TMenuItem', iptrw);
20989:     PSImport_CoObj1', 'TPSImport_CoObj', iptrw);
20990:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
20991:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
20992:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
20993:     tutorial4', 'TMenuItem', iptrw);
20994:     ExporttoClipboard1', 'TMenuItem', iptrw);
20995:     ImportfromClipboard1', 'TMenuItem', iptrw);
20996:     N4', 'TMenuItem', iptrw);
20997:     N5', 'TMenuItem', iptrw);
20998:     N6', 'TMenuItem', iptrw);
20999:     ImportfromClipboard2', 'TMenuItem', iptrw);
21000:     tutorial11', 'TMenuItem', iptrw);
21001:     N7', 'TMenuItem', iptrw);
21002:     ShowSpecChars1', 'TMenuItem', iptrw);
21003:     OpenDirectory1', 'TMenuItem', iptrw);
21004:     procMess', 'TMenuItem', iptrw);
21005:     btnUseCase', 'TToolButton', iptrw);
21006:     ToolButton7', 'TToolButton', iptrw);
21007:     EditFont1', 'TMenuItem', iptrw);
21008:     UseCase1', 'TMenuItem', iptrw);
21009:     tutorial21', 'TMenuItem', iptrw);
21010:     OpenUseCase1', 'TMenuItem', iptrw);
21011:     PSImport_DB1', 'TPSImport_DB', iptrw);
21012:     tutorial31', 'TMenuItem', iptrw);
21013:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
21014:     HTMLSyntax1', 'TMenuItem', iptrw);
21015:     ShowInterfaces1', 'TMenuItem', iptrw);
21016:     Tutorials5', 'TMenuItem', iptrw);
21017:     AllFunctionsList1', 'TMenuItem', iptrw);
21018:     ShowLastException1', 'TMenuItem', iptrw);
21019:     PlayMP31', 'TMenuItem', iptrw);
21020:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
21021:     texSyntax1', 'TMenuItem', iptrw);
21022:     N8', 'TMenuItem', iptrw);
21023:     GetEMails1', 'TMenuItem', iptrw);
21024:     SynCppSyn1', 'TSynCppSyn', iptrw);
21025:     CSyntax1', 'TMenuItem', iptrw);
21026:     Tutorial6', 'TMenuItem', iptrw);
21027:     New1', 'TMenuItem', iptrw);
21028:     AllObjectsList1', 'TMenuItem', iptrw);
21029:     LoadBytecode1', 'TMenuItem', iptrw);
21030:     CipherFile1', 'TMenuItem', iptrw);
21031:     N9', 'TMenuItem', iptrw);
21032:     N10', 'TMenuItem', iptrw);
21033:     Tutorial11', 'TMenuItem', iptrw);
21034:     Tutorial71', 'TMenuItem', iptrw);
21035:     UpdateService1', 'TMenuItem', iptrw);
21036:     PascalSchool1', 'TMenuItem', iptrw);
21037:     Tutorial81', 'TMenuItem', iptrw);
21038:     DelphiSite1', 'TMenuItem', iptrw);
21039:     Output1', 'TMenuItem', iptrw);
21040:     TerminalStyle1', 'TMenuItem', iptrw);
21041:     ReadOnly1', 'TMenuItem', iptrw);
21042:     ShellStyle1', 'TMenuItem', iptrw);
21043:     BigScreen1', 'TMenuItem', iptrw);
```

```
21044: Tutorial91', 'TMenuItem', iptrw);
21045: SaveOutput2', 'TMenuItem', iptrw);
21046: N11', 'TMenuItem', iptrw);
21047: SaveScreenshot', 'TMenuItem', iptrw);
21048: Tutorial101', 'TMenuItem', iptrw);
21049: SQLSyntax1', 'TMenuItem', iptrw);
21050: SynSQLSyn1', 'TSynSQLSyn', iptrw);
21051: Console1', 'TMenuItem', iptrw);
21052: SynXMLSyn1', 'TSynXMLSyn', iptrw);
21053: XMLSyntax1', 'TMenuItem', iptrw);
21054: ComponentCount1', 'TMenuItem', iptrw);
21055: NewInstance1', 'TMenuItem', iptrw);
21056: toolbtnTutorial', 'TToolButton', iptrw);
21057: Memory1', 'TMenuItem', iptrw);
21058: SynJavaSyn1', 'TSynJavaSyn', iptrw);
21059: JavaSyntax1', 'TMenuItem', iptrw);
21060: SyntaxCheck1', 'TMenuItem', iptrw);
21061: Tutorial10Statistics1', 'TMenuItem', iptrw);
21062: ScriptExplorer1', 'TMenuItem', iptrw);
21063: FormOutput1', 'TMenuItem', iptrw);
21064: ArduinoDump1', 'TMenuItem', iptrw);
21065: AndroidDump1', 'TMenuItem', iptrw);
21066: GotoEnd1', 'TMenuItem', iptrw);
21067: AllResourceList1', 'TMenuItem', iptrw);
21068: ToolButton4', 'TToolButton', iptrw);
21069: btn6res', 'TToolButton', iptrw);
21070: Tutorial11Forms1', 'TMenuItem', iptrw);
21071: Tutorial12SQL1', 'TMenuItem', iptrw);
21072: ResourceExplore1', 'TMenuItem', iptrw);
21073: Info1', 'TMenuItem', iptrw);
21074: N12', 'TMenuItem', iptrw);
21075: CryptoBox1', 'TMenuItem', iptrw);
21076: Tutorial13Ciphering1', 'TMenuItem', iptrw);
21077: CipherFile2', 'TMenuItem', iptrw);
21078: N13', 'TMenuItem', iptrw);
21079: ModulesCount1', 'TMenuItem', iptrw);
21080: AddOns2', 'TMenuItem', iptrw);
21081: N4GewinntGame1', 'TMenuItem', iptrw);
21082: DocuforAddOns1', 'TMenuItem', iptrw);
21083: Tutorial14Async1', 'TMenuItem', iptrw);
21084: Lessons15Review1', 'TMenuItem', iptrw);
21085: SynPHPSyn1', 'TSynPHPSyn', iptrw);
21086: PHPSyntax1', 'TMenuItem', iptrw);
21087: Breakpoint1', 'TMenuItem', iptrw);
21088: SerialRS2321', 'TMenuItem', iptrw);
21089: N14', 'TMenuItem', iptrw);
21090: SyncCSSyn1', 'TSynCSSyn', iptrw);
21091: CSyntax2', 'TMenuItem', iptrw);
21092: Calculator1', 'TMenuItem', iptrw);
21093: btnSerial', 'TToolButton', iptrw);
21094: ToolButton8', 'TToolButton', iptrw);
21095: Tutorial151', 'TMenuItem', iptrw);
21096: N15', 'TMenuItem', iptrw);
21097: N16', 'TMenuItem', iptrw);
21098: ControlBar1', 'TControlBar', iptrw);
21099: ToolBar2', 'TToolBar', iptrw);
21100: BtnOpen', 'TToolButton', iptrw);
21101: BtnSave', 'TToolButton', iptrw);
21102: BtnPrint', 'TToolButton', iptrw);
21103: BtnColors', 'TToolButton', iptrw);
21104: btnClassReport', 'TToolButton', iptrw);
21105: BtnRotateRight', 'TToolButton', iptrw);
21106: BtnFullScreen', 'TToolButton', iptrw);
21107: BtnFitToWindowSize', 'TToolButton', iptrw);
21108: BtnZoomMinus', 'TToolButton', iptrw);
21109: BtnZoomPlus', 'TToolButton', iptrw);
21110: Panel1', 'TPanel', iptrw);
21111: LabelBrettgroesse', ' TLabel', iptrw);
21112: CB1SCList', 'TComboBox', iptrw);
21113: ImageListNormal', 'TImageList', iptrw);
21114: spbtnexpose', 'TSpeedButton', iptrw);
21115: spbtnexsample', 'TSpeedButton', iptrw);
21116: spbsaveas', 'TSpeedButton', iptrw);
21117: imglogobox', 'TImage', iptrw);
21118: EnlargeFont1', 'TMenuItem', iptrw);
21119: EnlargeFont2', 'TMenuItem', iptrw);
21120: ShrinkFont1', 'TMenuItem', iptrw);
21121: ThreadDemo1', 'TMenuItem', iptrw);
21122: HEXEditor1', 'TMenuItem', iptrw);
21123: HEXView1', 'TMenuItem', iptrw);
21124: HEXInspect1', 'TMenuItem', iptrw);
21125: SynExporterHTML1', 'TSynExporterHTML', iptrw);
21126: ExporttoHTML1', 'TMenuItem', iptrw);
21127: ClassCount1', 'TMenuItem', iptrw);
21128: HTMLOutput1', 'TMenuItem', iptrw);
21129: HEXEditor2', 'TMenuItem', iptrw);
21130: Minesweeper1', 'TMenuItem', iptrw);
21131: N17', 'TMenuItem', iptrw);
21132: PicturePuzzle1', 'TMenuItem', iptrw);
```

```

21133: sbvc1help', 'TSpeedButton', iptrw);
21134: DependencyWalker1', 'TMenuItem', iptrw);
21135: WebScanner1', 'TMenuItem', iptrw);
21136: View1', 'TMenuItem', iptrw);
21137: mnToolbar1', 'TMenuItem', iptrw);
21138: mnStatusbar2', 'TMenuItem', iptrw);
21139: mnConsole2', 'TMenuItem', iptrw);
21140: mnCoolbar2', 'TMenuItem', iptrw);
21141: mnSplitter2', 'TMenuItem', iptrw);
21142: WebServer1', 'TMenuItem', iptrw);
21143: Tutorial17Server1', 'TMenuItem', iptrw);
21144: Tutorial18Arduino1', 'TMenuItem', iptrw);
21145: SynPerlSyn1', 'TSynPerlSyn', iptrw);
21146: PerlSyntax1', 'TMenuItem', iptrw);
21147: SynPythonSyn1', 'TSynPythonSyn', iptrw);
21148: PythonSyntax1', 'TMenuItem', iptrw);
21149: DMathLibrary1', 'TMenuItem', iptrw);
21150: IntfNavigator1', 'TMenuItem', iptrw);
21151: EnlargeFontConsole1', 'TMenuItem', iptrw);
21152: ShrinkFontConsole1', 'TMenuItem', iptrw);
21153: SetInterfaceList1', 'TMenuItem', iptrw);
21154: popintfList', 'TPopupMenu', iptrw);
21155: intfAdd1', 'TMenuItem', iptrw);
21156: intfDelete1', 'TMenuItem', iptrw);
21157: intfRefactor1', 'TMenuItem', iptrw);
21158: Defactor1', 'TMenuItem', iptrw);
21159: Tutorial19COMArduino1', 'TMenuItem', iptrw);
21160: Tutorial20Regex', 'TMenuItem', iptrw);
21161: N18', 'TMenuItem', iptrw);
21162: Manual1', 'TMenuItem', iptrw);
21163: FullTextFinder1', 'TMenuItem', iptrw);
21164: Move1', 'TMenuItem', iptrw);
21165: FractalDemo1', 'TMenuItem', iptrw);
21166: Tutorial21Android1', 'TMenuItem', iptrw);
21167: Tutorial0Function1', 'TMenuItem', iptrw);
21168: SimuLogBox1', 'TMenuItem', iptrw);
21169: OpenExamples1', 'TMenuItem', iptrw);
21170: SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
21171: JavaScriptSyntax1', 'TMenuItem', iptrw);
21172: Halt1', 'TMenuItem', iptrw);
21173: CodeSearch1', 'TMenuItem', iptrw);
21174: SynRubySyn1', 'TSynRubySyn', iptrw);
21175: RubySyntax1', 'TMenuItem', iptrw);
21176: Undo1', 'TMenuItem', iptrw);
21177: SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
21178: LinuxShellScript1', 'TMenuItem', iptrw);
21179: Rename1', 'TMenuItem', iptrw);
21180: spdcodesearch', 'TSpeedButton', iptrw);
21181: Preview1', 'TMenuItem', iptrw);
21182: Tutorial22Services1', 'TMenuItem', iptrw);
21183: Tutorial23RealTime1', 'TMenuItem', iptrw);
21184: Configuration1', 'TMenuItem', iptrw);
21185: MP3Player1', 'TMenuItem', iptrw);
21186: DLLSpy1', 'TMenuItem', iptrw);
21187: SynURIOpener1', 'TSynURIOpener', iptrw);
21188: SynURISyn1', 'TSynURISyn', iptrw);
21189: URILinksClicks1', 'TMenuItem', iptrw);
21190: EditReplace1', 'TMenuItem', iptrw);
21191: GotoLine1', 'TMenuItem', iptrw);
21192: ActiveLineColor1', 'TMenuItem', iptrw);
21193: ConfigFile1', 'TMenuItem', iptrw);
21194: SortIntlList', 'TMenuItem', iptrw);
21195: Redo1', 'TMenuItem', iptrw);
21196: Tutorial24CleanCode1', 'TMenuItem', iptrw);
21197: Tutorial25Configuration1', 'TMenuItem', iptrw);
21198: IndentSelection1', 'TMenuItem', iptrw);
21199: UnindentSection1', 'TMenuItem', iptrw);
21200: SkyStyle1', 'TMenuItem', iptrw);
21201: N19', 'TMenuItem', iptrw);
21202: CountWords1', 'TMenuItem', iptrw);
21203: imbookmarkimages', 'TImageList', iptrw);
21204: Bookmark11', 'TMenuItem', iptrw);
21205: N20', 'TMenuItem', iptrw);
21206: Bookmark21', 'TMenuItem', iptrw);
21207: Bookmark31', 'TMenuItem', iptrw);
21208: Bookmark41', 'TMenuItem', iptrw);
21209: SynMultiSyn1', 'TSynMultiSyn', iptrw);
21210:
21211: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
21212: Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSEExec; x:TPSRuntimeClassImporter);
21213: Procedure PSSScriptCompile( Sender : TPSScript)
21214: Procedure Compile1Click( Sender : TObject)
21215: Procedure PSSScriptExecute( Sender : TPSScript)
21216: Procedure open1Click( Sender : TObject)
21217: Procedure Save2Click( Sender : TObject)
21218: Procedure Savebefore1Click( Sender : TObject)
21219: Procedure Largefont1Click( Sender : TObject)
21220: Procedure FormActivate( Sender : TObject)
21221: Procedure SBytecode1Click( Sender : TObject)

```

```

21222: Procedure FormKeyPress( Sender : TObject; var Key : Char)
21223: Procedure Saveas3Click( Sender : TObject)
21224: Procedure Clear1Click( Sender : TObject)
21225: Procedure Slinenumbers1Click( Sender : TObject)
21226: Procedure About1Click( Sender : TObject)
21227: Procedure Search1Click( Sender : TObject)
21228: Procedure FormCreate( Sender : TObject)
21229: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
21230:                                         var Action : TSynReplaceAction)
21231: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
21232: Procedure WordWrap1Click( Sender : TObject)
21233: Procedure SearchNext1Click( Sender : TObject)
21234: Procedure Replace1Click( Sender : TObject)
21235: Function PSScriptNeedFile(Sender:TObject;const OriginFileName:String;var FName,Output:String):Bool;
21236: Procedure ShowInclude1Click( Sender : TObject)
21237: Procedure Printout1Click( Sender : TObject)
21238: Procedure mnuPrintFont1Click( Sender : TObject)
21239: Procedure Include1Click( Sender : TObject)
21240: Procedure FormDestroy( Sender : TObject)
21241: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
21242: Procedure UpdateViewClick( Sender : TObject)
21243: Procedure CodeCompletionList1Click( Sender : TObject)
21244: Procedure SaveOutput1Click( Sender : TObject)
21245: Procedure ExportClipboard1Click( Sender : TObject)
21246: Procedure Close1Click( Sender : TObject)
21247: Procedure Manual1Click( Sender : TObject)
21248: Procedure LoadLastFile1Click( Sender : TObject)
21249: Procedure Memo1Change( Sender : TObject)
21250: Procedure Decompile1Click( Sender : TObject)
21251: Procedure StepInto1Click( Sender : TObject)
21252: Procedure StepOut1Click( Sender : TObject)
21253: Procedure Reset1Click( Sender : TObject)
21254: Procedure cedebugAfterExecute( Sender : TPSScript)
21255: Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
21256: Procedure cedebugCompile( Sender : TPSScript)
21257: Procedure cedebugExecute( Sender : TPSScript)
21258: Procedure cedebugIdle( Sender : TObject)
21259: Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
21260: Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
21261: Procedure BreakPointMenuClick( Sender : TObject)
21262: Procedure DebugRun1Click( Sender : TObject)
21263: Procedure tutorial4Click( Sender : TObject)
21264: Procedure ImportfromClipboard1Click( Sender : TObject)
21265: Procedure ImportfromClipboard2Click( Sender : TObject)
21266: Procedure tutorial1Click( Sender : TObject)
21267: Procedure ShowSpecChars1Click( Sender : TObject)
21268: Procedure StatusBar1DblClick( Sender : TObject)
21269: Procedure PSScriptLine( Sender : TObject)
21270: Procedure Opendirectory1Click( Sender : TObject)
21271: Procedure procMessClick( Sender : TObject)
21272: Procedure tbtnUseCaseClick( Sender : TObject)
21273: Procedure EditFont1Click( Sender : TObject)
21274: Procedure tutorial21Click( Sender : TObject)
21275: Procedure tutorial31Click( Sender : TObject)
21276: Procedure HTMLEyntax1Click( Sender : TObject)
21277: Procedure ShowInterfaces1Click( Sender : TObject)
21278: Procedure Tutorial5Click( Sender : TObject)
21279: Procedure ShowLastException1Click( Sender : TObject)
21280: Procedure PlayMP31Click( Sender : TObject)
21281: Procedure AllFunctionsList1Click( Sender : TObject)
21282: Procedure texSyntax1Click( Sender : TObject)
21283: Procedure GetEMails1Click( Sender : TObject)
21284: procedure DelphiSite1Click(Sender: TObject);
21285: procedure TerminalStyle1Click(Sender: TObject);
21286: procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
21287: procedure Shellstyle1Click(Sender: TObject);
21288: procedure Console1Click(Sender: TObject); //3.2
21289: procedure BigScreen1Click(Sender: TObject);
21290: procedure Tutorial91Click(Sender: TObject);
21291: procedure SaveScreenshotClick(Sender: TObject);
21292: procedure Tutorial101Click(Sender: TObject);
21293: procedure SQLSyntax1Click(Sender: TObject);
21294: procedure XMLSyntax1Click(Sender: TObject);
21295: procedure ComponentCount1Click(Sender: TObject);
21296: procedure NewInstance1Click(Sender: TObject);
21297: procedure CSyntax1Click(Sender: TObject);
21298: procedure Tutorial6Click(Sender: TObject);
21299: procedure New1Click(Sender: TObject);
21300: procedure AllObjectsList1Click(Sender: TObject);
21301: procedure LoadBytecode1Click(Sender: TObject);
21302: procedure CipherFile1Click(Sender: TObject); //V3.5
21303: procedure NewInstance1Click(Sender: TObject);
21304: procedure toolbtnTutorialClick(Sender: TObject);
21305: procedure Memory1Click(Sender: TObject);
21306: procedure JavaSyntax1Click(Sender: TObject);
21307: procedure SyntaxCheck1Click(Sender: TObject);
21308: procedure ScriptExplorer1Click(Sender: TObject);
21309: procedure FormOutput1Click(Sender: TObject); //V3.6
21310: procedure GotoEnd1Click(Sender: TObject);

```

```

21311: procedure AllResourceList1Click(Sender: TObject);
21312: procedure tbtn6resClick(Sender: TObject); //V3.7
21313: procedure Info1Click(Sender: TObject);
21314: procedure Tutorial10Statistics1Click(Sender: TObject);
21315: procedure Tutorial11Forms1Click(Sender: TObject);
21316: procedure Tutorial12SQL1Click(Sender: TObject); //V3.8
21317: procedure ResourceExplore1Click(Sender: TObject);
21318: procedure Info1Click(Sender: TObject);
21319: procedure CryptoBox1Click(Sender: TObject);
21320: procedure ModulesCount1Click(Sender: TObject);
21321: procedure N4GewinntGame1Click(Sender: TObject);
21322: procedure PHPSyntax1Click(Sender: TObject);
21323: procedure SerialRS2321Click(Sender: TObject);
21324: procedure CSyntax2Click(Sender: TObject);
21325: procedure Calculator1Click(Sender: TObject);
21326: procedure Tutorial13Ciphering1Click(Sender: TObject);
21327: procedure Tutorial14Async1Click(Sender: TObject);
21328: procedure PHPSyntax1Click(Sender: TObject);
21329: procedure BtnZoomPlusClick(Sender: TObject);
21330: procedure BtnZoomMinusClick(Sender: TObject);
21331: procedure btnClassReportClick(Sender: TObject);
21332: procedure ThreadDemo1Click(Sender: TObject);
21333: procedure HEXView1Click(Sender: TObject);
21334: procedure ExporttoHTML1Click(Sender: TObject);
21335: procedure Minesweeper1Click(Sender: TObject);
21336: procedure PicturePuzzle1Click(Sender: TObject); //V3.9
21337: procedure sbvc1helpClick(Sender: TObject);
21338: procedure DependencyWalker1Click(Sender: TObject);
21339: procedure CBLISCLDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
21340: procedure WebScanner1Click(Sender: TObject);
21341: procedure mnToolbar1Click(Sender: TObject);
21342: procedure mnStatusbar2Click(Sender: TObject);
21343: procedure mnConsole2Click(Sender: TObject);
21344: procedure mnCoolbar2Click(Sender: TObject);
21345: procedure mnSplitter2Click(Sender: TObject);
21346: procedure WebServer1Click(Sender: TObject);
21347: procedure PerlSyntax1Click(Sender: TObject);
21348: procedure PythonSyntax1Click(Sender: TObject);
21349: procedure DMathLibrary1Click(Sender: TObject);
21350: procedure IntfNavigator1Click(Sender: TObject);
21351: procedure FullTextFinder1Click(Sender: TObject);
21352: function AppName: string;
21353: function ScriptName: string;
21354: function LastName: string;
21355: procedure FractalDemo1Click(Sender: TObject);
21356: procedure SimuLogBox1Click(Sender: TObject);
21357: procedure OpenExamples1Click(Sender: TObject);
21358: procedure Halt1Click(Sender: TObject);
21359: procedure Stop;
21360: procedure CodeSearch1Click(Sender: TObject);
21361: procedure RubySyntax1Click(Sender: TObject);
21362: procedure Undo1Click(Sender: TObject);
21363: procedure LinuxShellScript1Click(Sender: TObject);
21364: procedure WebScannerDirect(urls: string);
21365: procedure WebScanner(urls: string);
21366: procedure LoadInterfaceList2;
21367: procedure DLLSpy1Click(Sender: TObject);
21368: procedure Memo1DblClick(Sender: TObject);
21369: procedure URILinksClicks1Click(Sender: TObject);
21370: procedure GotoLine1Click(Sender: TObject);
21371: procedure ConfigFile1Click(Sender: TObject);
21372: Procedure Sort1IntlistClick( Sender : TObject )
21373: Procedure Redo1Click( Sender : TObject )
21374: Procedure Tutorial24CleanCode1Click( Sender : TObject )
21375: Procedure IndentSelection1Click( Sender : TObject )
21376: Procedure UnindentSection1Click( Sender : TObject )
21377: Procedure SkyStyle1Click( Sender : TObject )
21378: Procedure CountWords1Click( Sender : TObject )
21379: Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark )
21380: Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
21381: Procedure Bookmark11Click( Sender : TObject )
21382: Procedure Bookmark21Click( Sender : TObject )
21383: Procedure Bookmark31Click( Sender : TObject )
21384: Procedure Bookmark41Click( Sender : TObject )
21385: Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
21386: 'STATMemoryRepos', 'boolean', iptrw);
21387: 'IPPort', 'integer', iptrw);
21388: 'COMPort', 'integer', iptrw);
21389: '1bintlist', 'TListBox', iptrw);
21390: Function GetStatChange : boolean
21391: Procedure SetStatChange( vstat : boolean )
21392: Function GetActFileName : string
21393: Procedure SetActFileName( vname : string )
21394: Function GetLastFileName : string
21395: Procedure SetLastFileName( vname : string )
21396: Procedure WebScannerDirect( urls : string )
21397: Procedure LoadInterfaceList2
21398: Function GetStatExecuteShell : boolean
21399: Procedure DoEditorExecuteCommand( EditorCommand : word )

```

```

21400: function GetActiveLineColor: TColor
21401: procedure SetActiveLineColor(acolor: TColor)
21402: procedure ScriptListbox1Click(Sender: TObject);
21403: procedure Memo2KeyPress(Sender: TObject; var Key: Char);
21404: procedure EnlargeGutter1Click(Sender: TObject);
21405: procedure Tetris1Click(Sender: TObject);
21406: procedure ToDoList1Click(Sender: TObject);
21407: procedure ProcessList1Click(Sender: TObject);
21408: procedure MetricReport1Click(Sender: TObject);
21409: procedure ProcessList1Click(Sender: TObject);
21410: procedure TCPSockets1Click(Sender: TObject);
21411: procedure ConfigUpdate1Click(Sender: TObject);
21412: procedure ADOWorkbench1Click(Sender: TObject);
21413: procedure SocketServer1Click(Sender: TObject);
21414: procedure FormDemo1Click(Sender: TObject);
21415: procedure Richedit1Click(Sender: TObject);
21416: procedure SimpleBrowser1Click(Sender: TObject);
21417: procedure DOSShell1Click(Sender: TObject);
21418: procedure SynExport1Click(Sender: TObject);
21419: procedure ExporttoRTF1Click(Sender: TObject);
21420: procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
21421: procedure SOAPTester1Click(Sender: TObject);
21422: procedure Sniffer1Click(Sender: TObject);
21423: procedure AutoDetectSyntax1Click(Sender: TObject);
21424: procedure FPPlot1Click(Sender: TObject);
21425: procedure PassStyle1Click(Sender: TObject);
21426: procedure Tutorial183RGBLED1Click(Sender: TObject);
21427: procedure Reversi1Click(Sender: TObject);
21428: procedure Manualmaxbox1Click(Sender: TObject);
21429: procedure BlaisePascalMagazine1Click(Sender: TObject);
21430: procedure AddToDolClick(Sender: TObject);
21431: procedure CreateGUID1Click(Sender: TObject);
21432: procedure Tutorial27XML1Click(Sender: TObject);
21433: procedure CreateDLLStub1Click(Sender: TObject);
21434: procedure Tutorial28DLL1Click(Sender: TObject);');
21435: procedure ResetKeyPressed();';
21436: procedure KeyPressedFalse();
21437: procedure FileChanges1Click(Sender: TObject);';
21438: procedure OpenGLtry1Click(Sender: TObject);';
21439: procedure AllUnitList1Click(Sender: TObject);';
21440: procedure Tutorial29UMLClick(Sender: TObject);
21441: procedure CreateHeader1Click(Sender: TObject);
21442: procedure Oscilloscope1Click(Sender: TObject);';
21443: procedure Tutorial30WOT1Click(Sender: TObject);';
21444: procedure GetWebScript1Click(Sender: TObject);';
21445: procedure Checkers1Click(Sender: TObject);';
21446: procedure TaskMgr1Click(Sender: TObject);';
21447: procedure WebCam1Click(Sender: TObject);';
21448: procedure Tutorial31Closure1Click(Sender: TObject);';
21449: procedure GEOMapView1Click(Sender: TObject);';
21450: procedure Run1Click(Sender: TObject);
21451: MaxForm1.GPSSatView1Click, 'GPSSatView1Click');
21452: MaxForm1.N3DLab1Click, 'N3DLab1Click');
21453: procedure ExternalApp1Click(Sender: TObject);';
21454: procedure PANView1Click(Sender: TObject);
21455: procedure Tutorial39GEOMaps1Click(Sender: TObject);
21456: procedure UnitConverter1Click(Sender: TObject);
21457: //-----
21458: //*****mX4 Editor SynEdit Tools API *****
21459: //-----
21460: procedure SIRegister_TCustomSynEdit(CL: TSPSPascalCompiler);
21461: begin //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
21462:   with FindClass('TCustomControl','TCustomSynEdit') do begin
21463:     Constructor Create(AOwner : TComponent)
21464:     SelStart', 'Integer', iptrw);
21465:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
21466:     Procedure UpdateCaret
21467:     Procedure AddKey(Command: TSynEditorCommand; Key1: word; SS1: TShiftState; Key2: word; SS2: TShiftState);
21468:     Procedure AddKey(Command: TSynEditorCommand; Key1: word; SS1: TShiftState; Key2: word; SS2: TShiftState);
21469:     Procedure BeginUndoBlock
21470:     Procedure BeginUpdate
21471:       Function CaretInView : Boolean
21472:       Function CharIndexToRowCol( Index : integer ) : TBufferCoord
21473:       Procedure Clear
21474:       Procedure ClearAll
21475:       Procedure ClearBookMark( BookMark : Integer )
21476:       Procedure ClearSelection
21477:       Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer )
21478:       Procedure ClearUndo
21479:       Procedure CopyToClipboard
21480:       Procedure CutToClipboard
21481:       Procedure DoCopyToClipboard( const SText : string )
21482:       Procedure EndUndoBlock
21483:       Procedure EndUpdate
21484:       Procedure EnsureCursorPosVisible
21485:       Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean )
21486:       Procedure FindMatchingBracket
21487:       Function GetMatchingBracket : TBufferCoord
21488:       Function GetMatchingBracketEx( const APoint : TBufferCoord ) : TBufferCoord

```

```

21489: Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
21490: Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
21491: Function GetHighlighterAttriAtRowCol( const XY : TBufferCoord; var Token : string; var Attr
21492: : TSynHighlighterAttributes) : boolean
21493: Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
21494: var TokenType, Start : Integer; var Attr:TSynHighlighterAttributes):boolean
21495: Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
21496: Function GetWordAtRowCol( const XY : TBufferCoord) : string
21497: Procedure GotoBookMark( BookMark : Integer)
21498: Procedure GotoLineAndCenter( ALine : Integer)
21499: Function IdentChars : TSynIdentChars
21500: Procedure InvalidateGutter
21501: Procedure InvalidateGutterLine( aLine : integer)
21502: Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
21503: Procedure InvalidateLine( Line : integer)
21504: Procedure InvalidateLines( FirstLine, LastLine : integer)
21505: Procedure InvalidateSelection
21506: Function IsBookmark( BookMark : integer) : boolean
21507: Function IsPointInSelection( const Value : TBufferCoord) : boolean
21508: Procedure LockUndo
21509: Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
21510: Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
21511: Function LineToRow( aLine : integer) : integer
21512: Function RowToLine( aRow : integer) : integer
21513: Function NextWordPos : TBufferCoord
21514: Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
21515: Procedure PasteFromClipboard
21516: Function WordStart : TBufferCoord
21517: Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
21518: Function WordEnd : TBufferCoord
21519: Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
21520: Function PrevWordPos : TBufferCoord
21521: Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
21522: Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
21523: Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
21524: Procedure Redo
21525: Procedure RegisterCommandHandler( const AHandlerProc:THookedCommandEvent; AHandlerData:pointer)
21526: Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
21527: Function RowColToCharIndex( RowCol : TBufferCoord) : integer
21528: Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
21529: Procedure SelectAll
21530: Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
21531: Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
21532: Procedure SetDefaultKeystrokes
21533: Procedure SetSelWord
21534: Procedure SetWordBlock( Value : TBufferCoord)
21535: Procedure Undo
21536: Procedure UnlockUndo
21537: Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
21538: Procedure AddKeyUpHandler( aHandler : TKeyEvent)
21539: Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
21540: Procedure AddKeyDownHandler( aHandler : TKeyEvent)
21541: Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
21542: Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
21543: Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
21544: Procedure AddFocusControl( aControl : TWinControl)
21545: Procedure RemoveFocusControl( aControl : TWinControl)
21546: Procedure AddMouseDownHandler( aHandler : TMouseEvent)
21547: Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
21548: Procedure AddMouseUpHandler( aHandler : TMouseEvent)
21549: Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
21550: Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
21551: Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
21552: Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
21553: Procedure RemoveLinesPointer
21554: Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
21555: Procedure UnHookTextBuffer
21556: BlockBegin', 'TBufferCoord', iptrw);
21557: BlockEnd', 'TBufferCoord', iptrw);
21558: CanPaste', 'Boolean', iptr);
21559: CanRedo', 'boolean', iptr);
21560: CanUndo', 'boolean', iptr);
21561: CaretX', 'Integer', iptrw);
21562: CaretY', 'Integer', iptrw);
21563: CaretXY', 'TBufferCoord', iptrw);
21564: ActiveLineColor', 'TColor', iptrw);
21565: DisplayX', 'Integer', iptr);
21566: DisplayY', 'Integer', iptr);
21567: DisplayXY', 'TDisplayCoord', iptr);
21568: DisplayLineCount', 'integer', iptr);
21569: CharsInWindow', 'Integer', iptr);
21570: CharWidth', 'integer', iptr);
21571: Font', 'TFont', iptrw);
21572: GutterWidth', 'Integer', iptr);
21573: Highlighter', 'TSynCustomHighlighter', iptrw);
21574: LeftChar', 'Integer', iptrw);
21575: LineHeight', 'integer', iptr);
21576: LinesInWindow', 'Integer', iptr);
21577: LineText', 'string', iptrw); Lines', 'TStrings', iptrw);

```

```

21578: Marks', 'TSynEditMarkList', iptr);
21579: MaxScrollWidth', 'integer', iptrw);
21580: Modified', 'Boolean', iptrw);
21581: PaintLock', 'Integer', iptr);
21582: ReadOnly', 'Boolean', iptrw);
21583: SearchEngine', 'TSynEditSearchCustom', iptrw);
21584: SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
21585: SelTabBlock', 'Boolean', iptr);
21586: SelTabLine', 'Boolean', iptr); SelText', 'string', iptrw);
21587: StateFlags', 'TSynStateFlags', iptr);
21588: Text', 'string', iptrw); TopLine', 'Integer', iptrw);
21589: WordAtCursor', 'string', iptr);
21590: WordAtMouse', 'string', iptr);
21591: UndoList', 'TSynEditUndoList', iptr);
21592: RedoList', 'TSynEditUndoList', iptr);
21593: OnProcessCommandEvent', 'TProcessCommandEvent', iptrw);
21594: BookMarkOptions', 'TSynBookMarkOpt', iptrw);
21595: BorderStyle', 'TSynBorderStyle', iptrw);
21596: ExtraLineSpacing', 'integer', iptrw);
21597: Gutter', 'TSynGutter', iptrw);
21598: HideSelection', 'boolean', iptrw);
21599: InsertCaret', 'TSynEditCaretType', iptrw);
21600: InsertMode', 'boolean', iptrw);
21601: IsScrolling', 'Boolean', iptr);
21602: Keystrokes', 'TSynEditKeyStrokes', iptrw);
21603: MaxUndo', 'Integer', iptrw);
21604: Options', 'TSynEditorOptions', iptrw);
21605: OverwriteCaret', 'TSynEditCaretType', iptrw);
21606: RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
21607: ScrollHintColor', 'TColor', iptrw);
21608: ScrollHintFormat', 'TScrollHintFormat', iptrw);
21609: ScrollBars', 'TScrollStyle', iptrw);
21610: SelectedColor', 'TSynSelectedColor', iptrw);
21611: SelectionMode', 'TSynSelectionMode', iptrw);
21612: ActiveSelectionMode', 'TSynSelectionMode', iptrw);
21613: TabWidth', 'integer', iptrw); WantReturns', 'boolean', iptrw);
21614: WantTabs', 'boolean', iptrw); WordWrap', 'boolean', iptrw);
21615: WordWrapGlyph', 'TSynGlyph', iptrw);
21616: OnChange', 'TNotifyEvent', iptrw);
21617: OnClearBookmark', 'TPlaceMarkEvent', iptrw);
21618: OnCommandProcessed', 'TProcessCommandEvent', iptrw);
21619: OnContextHelp', 'TContextHelpEvent', iptrw);
21620: OnDropFiles', 'TDropFilesEvent', iptrw);
21621: OnGutterClick', 'TGutterClickEvent', iptrw);
21622: OnGutterGetText', 'TGutterGetTextEvent', iptrw);
21623: OnGutterPaint', 'TGutterPaintEvent', iptrw);
21624: OnMouseCursor', 'TMouseCursorEvent', iptrw);
21625: OnPaint', 'TPaintEvent', iptrw);
21626: OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
21627: OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
21628: OnReplaceText', 'TReplaceTextEvent', iptrw);
21629: OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
21630: OnStatusChange', 'TStatusChangeEvent', iptrw);
21631: OnPaintTransient', 'TPaintTransient', iptrw);
21632: OnScroll', 'TScrollEvent', iptrw);
21633: end;
21634: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
21635: Function GetPlaceableHighlighters : TSynHighlighterList
21636: Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
21637: Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
21638: Procedure GetEditorCommandValues( Proc : TGetStrProc)
21639: Procedure GetEditorCommandExtended( Proc : TGetStrProc)
21640: Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
21641: Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
21642: Function ConvertCodeStringToExtended( AString : String) : String
21643: Function ConvertExtendedToCodeString( AString : String) : String
21644: Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
21645: Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
21646: Function IndexToEditorCommand( const AIndex : Integer) : Integer
21647:
21648: TSynEditorOption = (
21649:   eoAltSetsColumnMode,           //Holding down the Alt Key will put the selection mode into columnar format
21650:   eoAutoIndent,                 //Will indent caret on newlines with same amount of leading whitespace as
21651:                           //preceding line
21652:   eoAutoSizeMaxScrollWidth,     //Automatically resizes the MaxScrollWidth property when inserting text
21653:   eoDisableScrollArrows,       //Disables the scroll bar arrow buttons when you can't scroll in that
21654:                           //direction any more
21655:   eoDragDropEditing,           //Allows to select a textblock and drag it in document to another location
21656:   eoDropFiles,                  //Allows the editor accept OLE file drops
21657:   eoEnhanceHomeKey,            //enhances home key positioning, similar to visual studio
21658:   eoEnhanceEndKey,             //enhances End key positioning, similar to JDeveloper
21659:   eoGroupUndo,                  //When undoing/redoing actions, handle all cont.changes same kind in onecall
21660:                           //instead undoing/redoing each command separately
21661:   eoHalfPageScroll,             //By scrolling with page-up/page-down commands, only scroll half page at time
21662:   eoHideShowScrollbars,        //if enabled, then scrollbars will only show if necessary.
21663:   If you have ScrollPasteEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
21664:   eoKeepCaretX,                //When moving through lines w/o cursor Past EOL, keeps X position of cursor
21665:   eoNoCaret,                   //Makes it so the caret is never visible
21666:   eoNoSelection,               //Disables selecting text

```

```

21667: eoRightMouseMovesCursor, //When clicking with right mouse for popup menu, moves cursor to location
21668: eoScrollByOneLess, //Forces scrolling to be one less
21669: eoScrollHintFollows, //The scroll hint follows the mouse when scrolling vertically
21670: eoScrollPastEof, //Allows the cursor to go past the end of file marker
21671: eoScrollPastEol, //Allows cursor to go past last character into white space at end of a line
21672: eoShowScrollHint, //Shows a hint of the visible line numbers when scrolling vertically
21673: eoShowSpecialChars, //Shows the special Characters
21674: eoSmartTabDelete, //similar to Smart Tabs, but when you delete characters
21675: eoSmartTabs, //When tabbing, cursor will go to non-white space character of previous line
21676: eoSpecialLineDefaultFg, //disables the foreground text color override using OnSpecialLineColor event
21677: eoTabIndent, //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
21678: eoTabsToSpaces, //Converts a tab character to a specified number of space characters
21679: eoTrimTrailingSpaces //Spaces at the end of lines will be trimmed and not saved
21680:
21681: *****Important Editor Short Cuts*****
21682: Double click to select a word and count words with lightning.
21683: Triple click to select a line.
21684: CTRL+SHIFT+click to extend a selection.
21685: Drag with the ALT key down to select columns of text !!!
21686: Drag and drop is supported.
21687: Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
21688: Type CTRL+A to select all.
21689: Type CTRL+N to set a new line.
21690: Type CTRL+T to delete a line or token. //Tokenizer
21691: Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
21692: Type CTRL+Shift+T to add ToDo in line and list.
21693: Type CTRL+Shift+[0..9] to set bookmarks. //Bookmark
21694: Type CTRL[0..9] to jump or get to bookmarks.
21695: Type Home to position cursor at beginning of current line and End to position it at end of line.
21696: Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
21697: Page Up and Page Down work as expected.
21698: CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
21699: using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
21700:
21701: {$ Short Key Positions Ctrl<A-Z>: }
21702: def
21703: <A> Select All
21704: <B> Count Words
21705: <C> Copy
21706: <D> Internet Start
21707: <E> Script List
21708: <F> Find
21709: <G> Goto
21710: <H> Mark Line
21711: <I> Interface List
21712: <J> Code Completion
21713: <K> Console
21714: <L> Interface List Box
21715: <M> Font Smaller -
21716: <N> New Line
21717: <O> Open File
21718: <P> Font Larger +
21719: <Q> Quit
21720: <R> Replace
21721: <S> Save!
21722: <T> Delete Line
21723: <U> Use Case Editor
21724: <V> Paste
21725: <W> URI Links
21726: <X> Reserved for coding use internal
21727: <Y> Delete Line
21728: <Z> Undo
21729:
21730: ref F1 Help
21731: F2 Syntax Check
21732: F3 Search Next
21733: F4 New Instance
21734: F5 Line Mark /Breakpoint
21735: F6 Goto End
21736: F7 Debug Step Into
21737: F8 Debug Step Out
21738: F9 Compile
21739: F10 Menu
21740: F11 Word Count Highlight
21741: F12 Reserved for coding use internal
21742:
21743: AddRegisteredVariable('Application', 'TApplication');
21744: AddRegisteredVariable('Screen', 'TScreen');
21745: AddRegisteredVariable('Self', 'TForm');
21746: AddRegisteredVariable('Memol', 'TSynMemo');
21747: AddRegisteredVariable('memo2', 'TMemo');
21748: AddRegisteredVariable('maxForm1', 'TMaxform1'); //!
21749: AddRegisteredVariable('debugout', 'Tdebugoutput'); //!
21750: AddRegisteredVariable('hlog', 'THotlog'); //!
21751: AddRegisteredVariable( it ,integer'); //for closure!!
21752: AddRegisteredVariable( sr ,string'); //for closure
21753: AddRegisteredVariable( bt ,boolean'); //for closure
21754: AddRegisteredVariable( ft ,double'); //for closure
21755: AddRegisteredVariable( srlist ,TStringlist'); //for closures

```

```

21756:
21757: def ReservedWords: array[0..86] of string =
21758:   ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
21759:    'constructor', 'default', 'destructor', 'disinterface', 'div', 'do',
21760:    'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
21761:    'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
21762:    'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
21763:    'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
21764:    'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
21765:    'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
21766:    'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
21767:    'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
21768:    'public', 'published', def.ref.using.typedef, memo1', 'memo2', 'doc', 'maxform1', 'it';
21769: AllowedChars: array[0..5] of string = ('(', ')', '[', ']', ',', ',', t1,t2,t3: boolean;
21770: //-----
21771: //*****End of mx4 Public Tools API *****
21772: //-----
21773:
21774: maxbox Internal Inventory
21775: Amount of Functions: 14417
21776: Amount of Procedures: 8976
21777: Amount of Constructors: 1458
21778: Totals of Calls: 24851
21779: SHA1: of Win 3.9.9.190 A5C1A64E94EBC01D01B67DB35C3E27E50607DFFF
21780:
21781: ****
21782: Doc Short Manual with 50 Tips!
21783: ****
21784: - Install: just save your maxboxdef.ini before and then extract the zip file!
21785: - Toolbar: Click on the red maxbox Sign (right on top) opens your work directory or jump to <Help>
21786: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
21787: - Menu: With <Ctrl><F3> you can search for code on examples
21788: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
21789: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
21790: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
21791:
21792: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
21793: - Inifile: Refresh (reload) the inifile after edit with ..//Help/Config Update
21794: - Context Menu: You can printout your scripts as a pdf-file or html-export
21795: - Context: You do have a context menu with the right mouse click
21796:
21797: - Menu: With the UseCase Editor you can convert graphic formats too.
21798: - Menu: On menu Options you find Addons as compiled scripts
21799: - IDE: Menu Program: Run Only is faster, after F2 - You don't need a mouse use shortcuts
21800: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
21801: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
21802: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
21803:   or ttimer (you type classp and then CTRL J), or you type tstringlist and <Ctrl><J>
21804:
21805: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
21806: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
21807: - Code: If you code a loop till key-pressed use function: isKeyPressed;
21808: - Code: Macro set the macros #name,, #paAdministrator, #file,startmaxbox_extract_funcList399.txtter25.pdf
21809: - Code: change Syntax in autoreboot macro 'maxbootscript.txt'
21810: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
21811:   to delete and Click and mark to drag a bookmark
21812: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
21813: - IDE: A file info with system and script information you find in menu Program/Information
21814: - IDE: After change the config file in help you can update changes in menu Help/Config Update
21815: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
21816: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
21817: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
21818: - Editor: Set Bookmarks to check your work in app or code
21819: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
21820: - Editor: With //TODO: some description or DONE you set code entries for ToDo List in ..//Help/ToDo List
21821: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
21822: - IDE with menu /Options/ADO SQL Workbench you can manage your Database
21823: - Context Menu: You can write your docs with RichEdit RTF printout /Editor Form Options/Richedit
21824: - Menu: Set Interface Naviagator also with toogle <Ctrl L> or /View/Intf Navigator
21825: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
21826: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
21827: - Code Editor: Compile with <F9> but also Alt C in case <F9> isn't available;
21828: - Code: if you cant run a function try the second one, for ex. Voice() - Voice2(), inc() - incl()
21829: - IDE set bookmarks with <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
21830: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
21831: - IDE menu /Help/Tools/ open the Task Manager
21832:
21833: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
21834: - Add on when no browser is available start /Options/Add ons/Easy Brower
21835: - Add on SOAP Tester with SOP POST File
21836: - Add on IP Protocol Sniffer with List View
21837: - Add on OpenGL mX Robot Demo for android
21838: - Add on Checkers Game, Add on Oscilloscope /View GEO Map View3
21839:
21840: - Menu: Help/Tools as a Tool Section with DOS Opener
21841: - Menu Editor: export the code as RTF File
21842: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
21843: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
21844: - Context: Auto Detect of Syntax depending on file extension

```

```

21845: - Code: some Windows API function start with w in the name like wGetAtomName();
21846: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
21847: - IDE File Check with menu ..View/File Changes/...
21848: - Context: Create a Header with Create Header in Navigator List at right window
21849: - Code: use SysErrorMessage to get a real Error Description, Ex.
21850:   RemoveDir('c:\NoSuchFolder'); writeln('System Error Message:' + SysErrorMessage(GetLastError));
21851: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
21852: - Editor: with <Ctrl W> you can click on hyperlinks in Code - CTRL Click on link
21853: - Editor: with <Ctrl+Alt+R> you can write in RTF format with RichEdit link
21854: - Menu: Check Help/Tools! you can use richedit, DOS Shell or Explorer
21855:
21856: - using DLL example in maXbox: //function: {*****}
21857:   Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
21858:                                     cb: DWORD): BOOL; //stdcall;
21859:   External 'GetProcessMemoryInfo@psapi.dll stdcall';
21860:   Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
21861:   External 'OpenProcess@kernel32.dll stdcall';
21862:
21863: GCC Compile Ex Script
21864: procedure TFormMain_btnCompileClick(Sender: TObject);
21865: begin
21866:   AProcess:= TProcess.Create(Nil);
21867:   try AProcess.Commandline := 'gcc.exe "' + OpenDialog1.FileName + '"';
21868:   +' -o "' + OpenDialog2.FileName + '"';
21869:   AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
21870:   AProcess.Execute;
21871:   Memo2.Lines.BeginUpdate;
21872:   Memo2.Lines.Clear;
21873:   Memo2.Lines.LoadFromStream(AProcess.Output);
21874:   Memo2.Lines.EndUpdate;
21875: finally
21876:   AProcess.Free;
21877: end;
21878: end;
21879:
21880: ref Stopwatch pattern snip
21881: Time1:= Time;
21882: writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
21883: if initAndStartBoard then
21884:   writeln('Filesize: '+inttostr(filesize(FILESAVE)));
21885: writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
21886: PrintF('%d %s',[Trunc((Time-Time1)*24),
21887:                   FormatDateTime('h runtime:" nn:ss:zzz',Time-Time1)]);
21888:
21889: POST git-receive-pack (chunked)
21890: Pushing to https://github.com/maxkleiner/maXbox3.git
21891: To https://github.com/maxkleiner/maXbox3.git f127d21..c6a98da masterbox2 -> masterbox2
21892: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
21893:
21894: History Shell Hell - Walk the Talk
21895: PCT Precompile Technology , mX4 ScriptStudio
21896: Indy, JCL, Jedi, VCL, Sysutils, TurboPower, Fundamentals, ExtendedRTL, Synedit
21897: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
21898: emax layers: system-package-component-unit-class-function-block
21899: new keywords def ref using maXCalcF UML: use case act class state seq pac comp dep - lib lab
21900: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
21901: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
21902: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
21903: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
21904: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
21905: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
21906: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
21907: SendBuffer, Color+Caption hack, ComboSet, SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
21908: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
21909: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
21910: TFFixedCriticalSection, XPlatform beta, GCC Command Pipe
21911: Inno Install and Setup Routines Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
21912: TFFixedCriticalSection, XPlatform beta, GCC Command Pipe
21913: VFW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
21914: 9 Color LED, LED Resources, Runtime LED, it + sr var, morse generator
21915: Add 5 Units, 1 Tutors, maxmap, OpenStreetView, MAPX
21916: Functions Menu/View/GEO Map View, DownloadFile, wgetX, sensors
21917: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
21918: ByteCode2, IPUtils2, GEOCode, CGI-Powtils, GPS_2, External App, Unit Converter
21919: Add 16 Units, 1 Slide, Tutor, Big Numbers (Decimals, TInteger), ModBusTCP, TGEOInfo
21920: Add 12 Units, 1 Tutor, SOAPConn, AVI Res, OLEUtils
21921:
21922: Ref:
21923: https://unibe-ch.academia.edu/MaxKleiner
21924: http://www.slideshare.net/maxkleiner1
21925: http://www.scribd.com/max_kleiner
21926: http://www.delphiforfun.org/Programs/Utilities/index.htm
21927: http://www.slideshare.net/maxkleiner1
21928: http://s3.amazonaws.com/PreviewLinks/22959.html
21929: http://www.softwareschule.ch/arduino_training.pdf
21930: http://www.jrsoftware.org/isinfo.php
21931: http://www.be-precision.com/products/precision-builder/express/
21932: http://www.blaisepascal.eu/
21933: http://www.delphibasics.co.uk/

```

```

21934: http://www.youtube.com/watch?v=av89HAbqAsI
21935: http://www.angelfire.com/hi5/delphizeus/modal.html
21936: http://www.retroarchive.org/garbo/pc/turbopas/index.html
21937: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
21938: http://delphi.org/2014/01/every-android-api-for-delphi/
21939: https://en.wikipedia.org/wiki/User:Maxkleiner
21940: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
21941: https://bitbucket.org/max_kleiner/maxbox3
21942: https://bitbucket.org/max_kleiner/maxbox3/downloads
21943: https://bitbucket.org/max_kleiner/maxbox3/wiki/maxbox%20Tutorials
21944: http://www.slideshare.net/maxkleiner1/codereview-topics
21945:
21946: UrlGoogleQrCode='http://chart apis.google.com/chart?chs=%dx%d&cht=qr&chl=%s&chl=%s';
21947: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
21948: =renderBasicSearchNarrative&q=%s';
21949: UrlMapQuestAPIReverse='http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
21950: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
21951:
21952: function OpenMap(const Data: string): boolean;
21953: var encURL: string;
21954: begin
21955:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPPEncode(Data)]);
21956:   try //HttpGet(EncodedURL, mapStream); //WinInet
21957:     Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
21958:     //OpenDoc(Exepath+'openmapx.html');
21959:     S_ShellExecute(Exepath+'openmapx.html','','seCmdOpen');
21960:   finally
21961:     encURL:= '';
21962:   end;
21963: end;
21964:
21965: procedure GetGEOMap(C_form,apath: string; const Data: string);
21966: var encodedURL: string; mapStream: TMemoryStream;
21967: begin //encodedURL:= Format(UrlGoogleQrCode,[Width,Height, C_Level, HTTPPEncode(Data)]);
21968:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPPEncode(Data)]);
21969:   mapStream:= TMemoryStream.create;
21970:   try
21971:     Wininet_HttpGet(EncodedURL, mapStream); //WinInet
21972:     mapStream.Position:= 0;
21973:     mapStream.Savetofile(apath); // OpenDoc(apath);
21974:     S_ShellExecute(apath,'',seCmdOpen);
21975:   finally
21976:     mapStream.Free;
21977:   end;
21978: end;
21979:
21980: Procedure BtnFactory(a,b,c,d:smallint; title,apic:string;
21981:                         var abtn:TBitBtn; anEvent:TNotifyEvent; afrm:TForm);
21982: begin
21983:   abtn:= TBitBtn.create(afrm);
21984:   with abtn do begin
21985:     parent:= afrm;
21986:     setBounds(a,b,c,d)
21987:     font.size:= 12;
21988:     glyph.LoadFromResourceName(HINSTANCE, apic);
21989:     mXButton(5,5,width, height,12,12,handle);
21990:     caption:= title;
21991:     onClick:= anEvent As TNotifyEvent;
21992:   end;
21993: end;
21994:
21995: function MySoundcard: Longint; external 'waveOutGetNumDevs@winmm.dll stdcall';
21996: function isSound: boolean; begin result:= mySoundcard > 0 end;
21997:
21998: function ListIdentical2(l1,l2:TStringList): Boolean;
21999: begin Result:= False;
22000:   if l1.count = l2.count then begin
22001:     for it:= 0 to l1.count-1 do
22002:       if (l1[it] <> l2[it]) then Exit;
22003:     Result:= True;
22004:   end;
22005: end;
22006:
22007:
22008: ****
22009: unit List asm internal end
22010: ****
22011: 01 unit RIRegister_StrUtils_Routines(exec); //Delphi
22012: 02 unit SIRegister_IdStrings //Indy Sockets
22013: 03 unit RIRegister_niSTRING_Routines(Exec); //from RegEx
22014: 04 unit uPSI_fMain Functions; //maxbox Open Tools API
22015: 05 unit IFSI_WinFormlpuzzle; //maxbox
22016: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImagefileLibBCB
22017: 07 unit RegisterDateLibrary_R(exec); //Delphi
22018: 08 unit RIRegister_MathMax_Routines(exec); //Jedi & Delphi
22019: 09 unit RIRegister_IdGlobal_Routines(exec); //Indy Sockets
22020: 10 unit RIRegister_SysUtils_Routines(Exec); //Delphi
22021: 11 unit uPSI_IdTCPConnection; //Indy some functions
22022: 12 unit uPSCompiler.pas; //PS kernel functions

```

```

22023: 13 unit uPSI_DBCommon;                                //DB Common_Routines and Types
22024: 14 unit uPSI_Printers.pas                           //Delphi VCL
22025: 15 unit uPSI_MPlayer.pas                           //Delphi VCL
22026: 16 unit uPSC_comobj;                               //COM Functions
22027: 17 unit uPSI_Clipbrd;                             //Delphi VCL
22028: 18 unit Filectrl in IFSI_SysUtils_max;           //VCL Runtime
22029: 19 unit uPSI_SqlExpr;                            //DBX3
22030: 20 unit uPSI_ADOdb;                             //ADODB
22031: 21 unit uPSI_StrHlpr;                           //String Helper Routines
22032: 22 unit uPSI_DateUtils;                          //Expansion to Date/TimeLib
22033: 23 unit uPSI_FileUtils;                          //Expansion to Sys/File Utils
22034: 24 unit JUtils / gsUtils;                        //Jedi / Metabase
22035: 25 unit JvFunctions_max;                         //Jedi Functions
22036: 26 unit HTTPParser;                            //Delphi VCL
22037: 27 unit HTTPUtil;                             //Delphi VCL
22038: 28 unit uPSI_XMLUtil;                           //Delphi VCL
22039: 29 unit uPSI_SOAPHTTPClient;                   //Delphi VCL SOAP WebService V3.5
22040: 30 unit uPSI_Contrns;                           //Delphi RTL Container of Classes
22041: 31 unit uPSI_MaskUtils;                         //RTL Edit and Mask functions
22042: 32 unit uPSI_MyBigInt;                          //big integer class with Math
22043: 33 unit uPSI_ConvUtils;                         //Delphi VCL Conversions engine
22044: 34 unit Types_Variants;                        //Delphi\Win32\rtl\sys
22045: 35 unit uPSI_IdHashSHA1;                        //Indy Crypto Lib
22046: 36 unit uPSI_IdHashMessageDigest;              //Indy Crypto;
22047: 37 unit uPSI_IdASN1Util;                        //Indy ASN1Utility Routines;
22048: 38 unit uPSI_IdLogFile;                         //Indy Logger from LogBase
22049: 39 unit uPSI_IdICmpClient;                     //Indy Ping ICMP
22050: 40 unit uPSI_IdHashMessageDigest_max;          //Indy Crypto &OpenSSL;
22051: 41 unit uPSI_FileCtrl;                          //Delphi RTL
22052: 42 unit uPSI_Outline;                           //Delphi VCL
22053: 43 unit uPSI_ScktComp;                          //Delphi RTL
22054: 44 unit uPSI_Calendar;                         //Delphi VCL
22055: 45 unit uPSI_VListView;                         //VListView;
22056: 46 unit uPSI_DBGrids;                           //Delphi VCL
22057: 47 unit uPSI_DBCtrls;                           //Delphi VCL
22058: 48 unit ide_debugoutput;                       //maXbox
22059: 49 unit uPSI_ComCtrls;                          //Delphi VCL
22060: 50 unit uPSC_stdctrls+;                        //Delphi VCL
22061: 51 unit uPSI_Dialogs;                           //Delphi VCL
22062: 52 unit uPSI_StdConvs;                          //Delphi RTL
22063: 53 unit uPSI_DBClient;                          //Delphi RTL
22064: 54 unit uPSI_DBPlatform;                       //Delphi RTL
22065: 55 unit uPSI_Provider;                          //Delphi RTL
22066: 56 unit uPSI_FMTBcd;                           //Delphi RTL
22067: 57 unit uPSI_DBGrids;                           //Delphi VCL
22068: 58 unit uPSI_CDSTUtil;                          //MIDAS
22069: 59 unit uPSI_VarHlpr;                           //Delphi RTL
22070: 60 unit uPSI_ExtDlgs;                           //Delphi VCL
22071: 61 unit sdpStopwatch;                          //maXbox
22072: 62 unit uPSI_JclStatistics;                   //JCL
22073: 63 unit uPSI_JclLogic;                          //JCL
22074: 64 unit uPSI_JclMiscel;                        //JCL
22075: 65 unit uPSI_JclMath_max;                      //JCL RTL
22076: 66 unit uPSI_uTPLb_StreamUtils;               //LockBox 3
22077: 67 unit uPSI_MathUtils;                        //BCB
22078: 68 unit uPSI_JclMultimedia;                  //JCL
22079: 69 unit uPSI_WideStrUtils;                    //Delphi API/RTL
22080: 70 unit uPSI_GraphUtil;                         //Delphi RTL
22081: 71 unit uPSI_TypeTrans;                        //Delphi RTL
22082: 72 unit uPSI_HTTPApp;                          //Delphi VCL
22083: 73 unit uPSI_DBWeb;                            //Delphi VCL
22084: 74 unit uPSI_DBBdeWeb;                          //Delphi VCL
22085: 75 unit uPSI_DBXpressWeb;                     //Delphi VCL
22086: 76 unit uPSI_ShadowWnd;                        //Delphi VCL
22087: 77 unit uPSI_ToolWin;                           //Delphi VCL
22088: 78 unit uPSI_Tabs;                            //Delphi VCL
22089: 79 unit uPSI_JclGraphUtils;                   //JCL
22090: 80 unit uPSI_JclCounter;                        //JCL
22091: 81 unit uPSI_JclSysInfo;                      //JCL
22092: 82 unit uPSI_JclSecurity;                     //JCL
22093: 83 unit uPSI_JclFileUtils;                    //JCL
22094: 84 unit uPSI_IdUserAccounts;                  //Indy
22095: 85 unit uPSI_IdAuthentication;                //Indy
22096: 86 unit uPSI_uTPLb_AES;                        //LockBox 3
22097: 87 unit uPSI_IdHashSHA1;                      //LockBox 3
22098: 88 unit uTPLb_BlockCipher;                    //LockBox 3
22099: 89 unit uPSI_ValEdit.pas;                     //Delphi VCL
22100: 90 unit uPSI_JvVCLUtility;                   //JCL
22101: 91 unit uPSI_JvDBUtil;                         //JCL
22102: 92 unit uPSI_JvDBUtil;                         //JCL
22103: 93 unit uPSI_JvAppUtil;                        //JCL
22104: 94 unit uPSI_JvCtrlUtil;                       //JCL
22105: 95 unit uPSI_JvFormToHtml;                    //JCL
22106: 96 unit uPSI_JvParsing;                        //JCL
22107: 97 unit uPSI_SerDlg;                           //Toolbox
22108: 98 unit uPSI_Serial;                           //Toolbox
22109: 99 unit uPSI_JvComponent;                     //JCL
22110: 100 unit uPSI_JvCalc;                          //JCL
22111: 101 unit uPSI_JvBdeUtil;                      //JCL

```

```

22112: 102 unit uPSI_JvDateUtil;                                //JCL
22113: 103 unit uPSI_JvGenetic;                               //JCL
22114: 104 unit uPSI_JclBase;                                 //JCL
22115: 105 unit uPSI_JvUtils;                                 //JCL
22116: 106 unit uPSI_JvStrUtil;                               //JCL
22117: 107 unit uPSI_JvStrUtils;                             //JCL
22118: 108 unit uPSI_JvFileUtil;                             //JCL
22119: 109 unit uPSI_JvMemoryInfos;                          //JCL
22120: 110 unit uPSI_JvComputerInfo;                         //JCL
22121: 111 unit uPSI_JvgCommClasses;                        //JCL
22122: 112 unit uPSI_JvgLogics;                            //JCL
22123: 113 unit uPSI_JvLED;                                 //JCL
22124: 114 unit uPSI_JvTurtle;                             //JCL
22125: 115 unit uPSI_SortThds; unit uPSI_ThSort;           //maxbox
22126: 116 unit uPSI_JvgUtils;                            //JCL
22127: 117 unit uPSI_JvExprParser;                          //JCL
22128: 118 unit uPSI_HexDump;                             //Borland
22129: 119 unit uPSI_DBLogDlg;                           //VCL
22130: 120 unit uPSI_SqlTimSt;                           //RTL
22131: 121 unit uPSI_JvHtmlParser;                         //JCL
22132: 122 unit uPSI_JvgXMLSerializer;                     //JCL
22133: 123 unit uPSI_JvJCLUtils;                          //JCL
22134: 124 unit uPSI_JvStrings;                           //JCL
22135: 125 unit uPSI_uTPLb_IntegerUtils;                  //TurboPower
22136: 126 unit uPSI_uTPLb_HugeCardinal;                 //TurboPower
22137: 127 unit uPSI_uTPLb_HugeCardinalUtils;            //TurboPower
22138: 128 unit uPSI_SynRegExpr;                          //SynEdit
22139: 129 unit uPSI_StUtils;                            //SysTools4
22140: 130 unit uPSI_StToHTML;                           //SysTools4
22141: 131 unit uPSI_StStrms;                           //SysTools4
22142: 132 unit uPSI_StFIN;                            //SysTools4
22143: 133 unit uPSI_StAstroP;                          //SysTools4
22144: 134 unit uPSI_StStat;                            //SysTools4
22145: 135 unit uPSI_StNetCon;                          //SysTools4
22146: 136 unit uPSI_StDecMth;                          //SysTools4
22147: 137 unit uPSI_StOStr;                           //SysTools4
22148: 138 unit uPSI_StPtrns;                           //SysTools4
22149: 139 unit uPSI_StNetMsg;                          //SysTools4
22150: 140 unit uPSI_StMath;                            //SysTools4
22151: 141 unit uPSI_StExpEng;                          //SysTools4
22152: 142 unit uPSI_StCRC;                            //SysTools4
22153: 143 unit uPSI_StExport;                           //SysTools4
22154: 144 unit uPSI_StExpLog;                          //SysTools4
22155: 145 unit uPSI_ActnList;                           //Delphi VCL
22156: 146 unit uPSI_jpeg;                             //Borland
22157: 147 unit uPSI_StRandom;                          //SysTools4
22158: 148 unit uPSI_StDict;                           //SysTools4
22159: 149 unit uPSI_StBCD;                            //SysTools4
22160: 150 unit uPSI_StTxtDat;                          //SysTools4
22161: 151 unit uPSI_StRegEx;                           //SysTools4
22162: 152 unit uPSI_IMouse;                           //VCL
22163: 153 unit uPSI_SyncObjs;                          //VCL
22164: 154 unit uPSI_AsyncCalls;                        //Hausladen
22165: 155 unit uPSI_ParallelJobs;                      //Saraiva
22166: 156 unit uPSI_Variants;                          //VCL
22167: 157 unit uPSI_VarCmplx;                          //VCL Wolfram
22168: 158 unit uPSI_DTDSchema;                         //VCL
22169: 159 unit uPSI_ShLwApi;                           //Brakel
22170: 160 unit uPSI_IBUtils;                           //VCL
22171: 161 unit uPSI_CheckLst;                           //VCL
22172: 162 unit uPSI_JvSimpleXml;                        //JCL
22173: 163 unit uPSI_JclSimpleXml;                      //JCL
22174: 164 unit uPSI_JvXmlDatabase;                     //JCL
22175: 165 unit uPSI_JvMaxPixel;                         //JCL
22176: 166 unit uPSI_JvItemsSearchs;                    //JCL
22177: 167 unit uPSI_StExpEng2;                          //SysTools4
22178: 168 unit uPSI_StGenLog;                           //SysTools4
22179: 169 unit uPSI_JvLogFile;                          //Jcl
22180: 170 unit uPSI_CPort;                            //ComPort Lib v4.11
22181: 171 unit uPSI_CPortCtl;                          //ComPort
22182: 172 unit uPSI_CPortEsc;                          //ComPort
22183: 173 unit BarCodeScaner;                          //ComPort
22184: 174 unit uPSI_JvGraph;                           //JCL
22185: 175 unit uPSI_JvComCtrls;                        //JCL
22186: 176 unit uPSI_GUITesting;                        //D Unit
22187: 177 unit uPSI_JvFindFiles;                        //JCL
22188: 178 unit uPSI_StSystem;                          //SysTools4
22189: 179 unit uPSI_JvKeyboardStates;                  //JCL
22190: 180 unit uPSI_JvMail;                            //JCL
22191: 181 unit uPSI_JclConsole;                        //JCL
22192: 182 unit uPSI_JclLANMan;                         //JCL
22193: 183 unit uPSI_IdCustomHTTPServer;                //Indy
22194: 184 unit IdHTTPServer;                           //Indy
22195: 185 unit uPSI_IdTCPServer;                       //Indy
22196: 186 unit uPSI_IdSocketHandle;                   //Indy
22197: 187 unit uPSI_IdIOHandlerSocket;                 //Indy
22198: 188 unit IdIOHandler;                           //Indy
22199: 189 unit uPSI_cutils;                            //Bloodshed
22200: 190 unit uPSI_BoldUtils;                          //boldsoft

```

```

22201: 191 unit uPSI_IdSimpleServer;                                //Indy
22202: 192 unit uPSI_IdSSLOpenSSL;                                 //Indy
22203: 193 unit uPSI_IdMultipartFormData;                            //Indy
22204: 194 unit uPSI_SynURIOpener;                                 //SynEdit
22205: 195 unit uPSI_PerlRegEx;                                   //PCRE
22206: 196 unit uPSI_IdHeaderList;                                 //Indy
22207: 197 unit uPSI_StFirst;                                    //SysTools4
22208: 198 unit uPSI_JvCtrls;                                    //JCL
22209: 199 unit uPSI_IdTrivialFTPBase;                            //Indy
22210: 200 unit uPSI_IdTrivialFTP;                                //Indy
22211: 201 unit uPSI_IdUDPBase;                                  //Indy
22212: 202 unit uPSI_IdUDPClient;                                //Indy
22213: 203 unit uPSI_utypes;                                   //for DMath.DLL
22214: 204 unit uPSI_ShellAPI;                                 //Borland
22215: 205 unit uPSI_IdRemoteCMDClient;                           //Indy
22216: 206 unit uPSI_IdRemoteCMDServer;                           //Indy
22217: 207 unit IdRexecServer;                                 //Indy
22218: 208 unit IdRexec; {unit uPSI_IdRexec;}                  //Indy
22219: 209 unit IdUDPServer;                                 //Indy
22220: 210 unit IdTimeUDPServer;                               //Indy
22221: 211 unit IdTimeServer;                                 //Indy
22222: 212 unit IdTimeUDP; {unit uPSI_IdUDPServer;}           //Indy
22223: 213 unit uPSI_IdIPWatch;                                //Indy
22224: 214 unit uPSI_IdIrcServer;                             //Indy
22225: 215 unit uPSI_IdMessageCollection;                      //Indy
22226: 216 unit uPSI_cPEM;                                    //Fundamentals 4
22227: 217 unit uPSI_cFundamentUtils;                          //Fundamentals 4
22228: 218 unit uPSI_uwinplot;                                //DMath
22229: 219 unit uPSI_xrtl_util_CPUUtils;                      //ExtendedRTL
22230: 220 unit uPSI_GR32_System;                             //Graphics32
22231: 221 unit uPSI_cFileutils;                            //Fundamentals 4
22232: 222 unit uPSI_cDateTime; {(timemachine)}             //Fundamentals 4
22233: 223 unit uPSI_cTimers; {(high precision timer)}     //Fundamentals 4
22234: 224 unit uPSI_cRandom;                                //Fundamentals 4
22235: 225 unit uPSI_ueval;                                 //DMath
22236: 226 unit uPSI_xrtl_net_URIUtils;                     //ExtendedRTL
22237: 227 unit xrtl_net_URIUtils;                           //ExtendedRTL
22238: 228 unit uPSI_ufft; {(FFT)}                         //DMath
22239: 229 unit uPSI_DBXChannel;                            //Delphi
22240: 230 unit uPSI_DBXIndyChannel;                         //Delphi Indy
22241: 231 unit uPSI_xrtl_util_COMCat;                      //ExtendedRTL
22242: 232 unit uPSI_xrtl_util_StrUtils;                   //ExtendedRTL
22243: 233 unit uPSI_xrtl_util_VariantUtils;               //ExtendedRTL
22244: 234 unit uPSI_xrtl_util_FileUtils;                 //ExtendedRTL
22245: 235 unit xrtl_util_Compat;                           //ExtendedRTL
22246: 236 unit uPSI_OleAuto;                                //Borland
22247: 237 unit uPSI_xrtl_util_COMUtils;                  //ExtendedRTL
22248: 238 unit uPSI_CmAdmCtl;                                //Borland
22249: 239 unit uPSI_ValEdit2;                                //VCL
22250: 240 unit uPSI_GR32; //Graphics32                    //Graphics32
22251: 241 unit uPSI_GR32_Image;                            //Graphics32
22252: 242 unit uPSI_xrtl_util_TimeUtils;                //ExtendedRTL
22253: 243 unit uPSI_xrtl_util_TimeZone;                 //ExtendedRTL
22254: 244 unit uPSI_xrtl_util_TimeStamp;                //ExtendedRTL
22255: 245 unit uPSI_xrtl_util_Map;                       //ExtendedRTL
22256: 246 unit uPSI_xrtl_util_Set;                      //ExtendedRTL
22257: 247 unit uPSI_CPortMonitor;                         //ComPort
22258: 248 unit uPSI_StInistm;                            //SysTools4
22259: 249 unit uPSI_GR32_ExtImage;                      //Graphics32
22260: 250 unit uPSI_GR32_OrdinalMaps;                  //Graphics32
22261: 251 unit uPSI_GR32_Rasterizers;                  //Graphics32
22262: 252 unit uPSI_xrtl_util_Exception;                //ExtendedRTL
22263: 253 unit uPSI_xrtl_util_Value;                   //ExtendedRTL
22264: 254 unit uPSI_xrtl_util_Compare;                //ExtendedRTL
22265: 255 unit uPSI_FlatSB;                                //VCL
22266: 256 unit uPSI_JvAnalogClock;                      //JCL
22267: 257 unit uPSI_JvAlarms;                            //JCL
22268: 258 unit uPSI_JvSQLS;                                //JCL
22269: 259 unit uPSI_JvDBSecur;                            //JCL
22270: 260 unit uPSI_JvDBQBE;                                //JCL
22271: 261 unit uPSI_JvStarfield;                           //JCL
22272: 262 unit uPSI_JVCLMiscal;                           //JCL
22273: 263 unit uPSI_JvProfiler32;                         //JCL
22274: 264 unit uPSI_JvDirectories;                        //JCL
22275: 265 unit uPSI_JclSchedule;                           //JCL
22276: 266 unit uPSI_JclSvcCtrl;                            //JCL
22277: 267 unit uPSI_JvSoundControl;                      //JCL
22278: 268 unit uPSI_JvBDESQLScript;                      //JCL
22279: 269 unit uPSI_JvgDigits;                            //JCL>
22280: 270 unit uPSI_ImgList;                                //TCustomImageList
22281: 271 unit uPSI_JclMIDI;                                //JCL>
22282: 272 unit uPSI_JclWinMidi;                            //JCL>
22283: 273 unit uPSI_JclNTFS;                                //JCL>
22284: 274 unit uPSI_JclAppInst;                            //JCL>
22285: 275 unit uPSI_JvRle;                                 //JCL>
22286: 276 unit uPSI_JvRas32;                                //JCL>
22287: 277 unit uPSI_JvImageDrawThread;                   //JCL>
22288: 278 unit uPSI_JvImageWindow;                          //JCL>
22289: 279 unit uPSI_JvTransparentForm;                   //JCL>

```

```

22290: 280 unit uPSI_JvWinDialogs; //JCL>
22291: 281 unit uPSI_JvSimLogic; //JCL>
22292: 282 unit uPSI_JvSimIndicator; //JCL>
22293: 283 unit uPSI_JvSimPID; //JCL>
22294: 284 unit uPSI_JvSimPIDLinker; //JCL>
22295: 285 unit uPSI_IdRFCReply; //Indy
22296: 286 unit uPSI_IdIdent; //Indy
22297: 287 unit uPSI_IdIdentServer; //Indy
22298: 288 unit uPSI_JvPatchFile; //JCL
22299: 289 unit uPSI_StNetPfm; //SysTools4
22300: 290 unit uPSI_StNet; //SysTools4
22301: 291 unit uPSI_JclPeImage; //JCL
22302: 292 unit uPSI_JclPrint; //JCL
22303: 293 unit uPSI_JclMime; //JCL
22304: 294 unit uPSI_JvRichEdit; //JCL
22305: 295 unit uPSI_JvDBRichEd; //JCL
22306: 296 unit uPSI_JvDice; //JCL
22307: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
22308: 298 unit uPSI_JvDirFrm; //JCL
22309: 299 unit uPSI_JvDualList; //JCL
22310: 300 unit uPSI_JvSwitch; //JCL
22311: 301 unit uPSI_JvTimerLst; //JCL
22312: 302 unit uPSI_JvMemTable; //JCL
22313: 303 unit uPSI_JvObjStr; //JCL
22314: 304 unit uPSI_StLArr; //SysTools4
22315: 305 unit uPSI_StWmDCpy; //SysTools4
22316: 306 unit uPSI_StText; //SysTools4
22317: 307 unit uPSI_StNTLog; //SysTools4
22318: 308 unit uPSI_xrtl_math_Integer; //ExtendedRTL
22319: 309 unit uPSI_JvImagPrvw; //JCL
22320: 310 unit uPSI_JvFormPatch; //JCL
22321: 311 unit uPSI_JvPicClip; //JCL
22322: 312 unit uPSI_JvDataConv; //JCL
22323: 313 unit uPSI_JvCpuUsage; //JCL
22324: 314 unit uPSI_JclUnitConv_mx2; //JCL
22325: 315 unit JvDualListForm; //JCL
22326: 316 unit uPSI_JvCpuUsage2; //JCL
22327: 317 unit uPSI_JvParserForm; //JCL
22328: 318 unit uPSI_JvJanTreeView; //JCL
22329: 319 unit uPSI_JvTransLED; //JCL
22330: 320 unit uPSI_JvPlaylist; //JCL
22331: 321 unit uPSI_JvFormAutoSize; //JCL
22332: 322 unit uPSI_JvYearGridEditForm; //JCL
22333: 323 unit uPSI_JvMarkupCommon; //JCL
22334: 324 unit uPSI_JvChart; //JCL
22335: 325 unit uPSI_JvXPCore; //JCL
22336: 326 unit uPSI_JvXPCoreUtils; //JCL
22337: 327 unit uPSI_StatsClasses; //mX4
22338: 328 unit uPSI_ExtCtrls2; //VCL
22339: 329 unit uPSI_JvUrlGrabbers; //JCL
22340: 330 unit uPSI_JvXmlTree; //JCL
22341: 331 unit uPSI_JvWavePlayer; //JCL
22342: 332 unit uPSI_JvUnicodeCanvas; //JCL
22343: 333 unit uPSI_JvTFUtils; //JCL
22344: 334 unit uPSI_IdServerIOHandler; //Indy
22345: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
22346: 336 unit uPSI_IdMessageCoder; //Indy
22347: 337 unit uPSI_IdMessageCoderMIME; //Indy
22348: 338 unit uPSI_IdMIMETypes; //Indy
22349: 339 unit uPSI_JvConverter; //JCL
22350: 340 unit uPSI_JvCsvParse; //JCL
22351: 341 unit uPSI_umath; unit uPSI_ugamma; //DMath
22352: 342 unit uPSI_ExcelExport; (Nat:TJsExcelExport) //JCL
22353: 343 unit uPSI_JvDBGridExport; //JCL
22354: 344 unit uPSI_JvgExport; //JCL
22355: 345 unit uPSI_JvSerialMaker; //JCL
22356: 346 unit uPSI_JvWin32; //JCL
22357: 347 unit uPSI_JvPaintFX; //JCL
22358: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
22359: 349 unit uPSI_JvValidators; (preview) //JCL
22360: 350 unit uPSI_JvNTEventLog; //JCL
22361: 351 unit uPSI_ShellZipTool; //mX4
22362: 352 unit uPSI_JvJoystick; //JCL
22363: 353 unit uPSI_JvMailSlots; //JCL
22364: 354 unit uPSI_JclComplex; //JCL
22365: 355 unit uPSI_SynPdf; //Synopse
22366: 356 unit uPSI_Registry; //VCL
22367: 357 unit uPSI_TlHelp32; //VCL
22368: 358 unit uPSI_JclRegistry; //JCL
22369: 359 unit uPSI_JvAirBrush; //JCL
22370: 360 unit uPSI_mORMotReport; //Synopse
22371: 361 unit uPSI_JclLocales; //JCL
22372: 362 unit uPSI_SynEdit; //SynEdit
22373: 363 unit uPSI_SynEditTypes; //SynEdit
22374: 364 unit uPSI_SynMacroRecorder; //SynEdit
22375: 365 unit uPSI_LongIntList; //SynEdit
22376: 366 unit uPSI_devutils; //DevC
22377: 367 unit uPSI_SynEditMiscClasses; //SynEdit
22378: 368 unit uPSI_SynEditRegexSearch; //SynEdit

```

```

22379: 369 unit uPSI_SynEditHighlighter; //SynEdit
22380: 370 unit uPSI_SynHighlighterPas; //SynEdit
22381: 371 unit uPSI_JvSearchFiles; //JCL
22382: 372 unit uPSI_SynHighlighterAny; //Lazarus
22383: 373 unit uPSI_SynEditKeyCmds; //SynEdit
22384: 374 unit uPSI_SynEditMiscProcs; //SynEdit
22385: 375 unit uPSI_SynEditKbdHandler; //SynEdit
22386: 376 unit uPSI_JvAppInst; //JCL
22387: 377 unit uPSI_JvAppEvent; //JCL
22388: 378 unit uPSI_JvAppCommand; //JCL
22389: 379 unit uPSI_JvAnimTitle; //JCL
22390: 380 unit uPSI_JvAnimatedImage; //JCL
22391: 381 unit uPSI_SynEditExport; //SynEdit
22392: 382 unit uPSI_SynExportHTML; //SynEdit
22393: 383 unit uPSI_SynExportRTF; //SynEdit
22394: 384 unit uPSI_SynEditSearch; //SynEdit
22395: 385 unit uPSI_fMain_back; //maxbox;
22396: 386 unit uPSI_JvZoom; //JCL
22397: 387 unit uPSI_PMrand; //PM
22398: 388 unit uPSI_JvSticker; //JCL
22399: 389 unit uPSI_XmlVerySimple; //mX4
22400: 390 unit uPSI_Services; //ExtPascal
22401: 391 unit uPSI_ExtPascalUtils; //ExtPascal
22402: 392 unit uPSI_SocketsDelphi; //ExtPascal
22403: 393 unit uPSI_StBarC; //SysTools
22404: 394 unit uPSI_StDbBarC; //SysTools
22405: 395 unit uPSI_StBarPN; //SysTools
22406: 396 unit uPSI_StDbPNBC; //SysTools
22407: 397 unit uPSI_StDb2DBC; //SysTools
22408: 398 unit uPSI_StMoney; //SysTools
22409: 399 unit uPSI_JvForth; //JCL
22410: 400 unit uPSI_RestRequest; //mX4
22411: 401 unit uPSI_HttpRESTConnectionIndy; //mX4
22412: 402 unit uPSI_JvXmlDatabase; //update //JCL
22413: 403 unit uPSI_StAstro; //SysTools
22414: 404 unit uPSI_StSort; //SysTools
22415: 405 unit uPSI_StDate; //SysTools
22416: 406 unit uPSI_StDateSt; //SysTools
22417: 407 unit uPSI_StBase; //SysTools
22418: 408 unit uPSI_StVInfo; //SysTools
22419: 409 unit uPSI_JvBrowseFolder; //JCL
22420: 410 unit uPSI_JvBoxProcs; //JCL
22421: 411 unit uPSI_urandom; //unit uranuvag; //DMath
22422: 412 unit uPSI_usimann; //unit ugenalg; //DMath
22423: 413 unit uPSI_JvHighlighter; //JCL
22424: 414 unit uPSI_Diff; //mX4
22425: 415 unit uPSI_SpringWinAPI; //DSpring
22426: 416 unit uPSI_StBits; //SysTools
22427: 417 unit uPSI_TomDBQue; //mX4
22428: 418 unit uPSI_MultilangTranslator; //mX4
22429: 419 unit uPSI_HyperLabel; //mX4
22430: 420 unit uPSI_Starter; //mX4
22431: 421 unit uPSI_FileAssoc; //devC
22432: 422 unit uPSI_devFileMonitorX; //devC
22433: 423 unit uPSI_devrunt; //devC
22434: 424 unit uPSI_devExec; //devC
22435: 425 unit uPSI_oyxUtils; //devC
22436: 426 unit uPSI_DosCommand; //devC
22437: 427 unit uPSI_CppTokenizer; //devC
22438: 428 unit uPSI_JvHLPParser; //devC
22439: 429 unit uPSI_JclMap; //JCL
22440: 430 unit uPSI_JclShell; //JCL
22441: 431 unit uPSI_JclCOM; //JCL
22442: 432 unit uPSI_GR32_Math; //Graphics32
22443: 433 unit uPSI_GR32_LowLevel; //Graphics32
22444: 434 unit uPSI_SimpleHl; //mX4
22445: 435 unit uPSI_GR32_Filters; //Graphics32
22446: 436 unit uPSI_GR32_VectorMaps; //Graphics32
22447: 437 unit uPSI_cXMLFunctions; //Fundamentals 4
22448: 438 unit uPSI_JvTTimer; //JCL
22449: 439 unit uPSI_cHTTPUtils; //Fundamentals 4
22450: 440 unit uPSI_cTLSUtils; //Fundamentals 4
22451: 441 unit uPSI_JclGraphics; //JCL
22452: 442 unit uPSI_JclSynch; //JCL
22453: 443 unit uPSI_IdTelnet; //Indy
22454: 444 unit uPSI_IdTelnetServer; //Indy
22455: 445 unit uPSI_IdEcho; //Indy
22456: 446 unit uPSI_IdEchoServer; //Indy
22457: 447 unit uPSI_IdEchoUDP; //Indy
22458: 448 unit uPSI_IdEchoUDPServer; //Indy
22459: 449 unit uPSI_IdSocks; //Indy
22460: 450 unit uPSI_IdAntiFreezeBase; //Indy
22461: 451 unit uPSI_IdHostnameServer; //Indy
22462: 452 unit uPSI_IdTunnelCommon; //Indy
22463: 453 unit uPSI_IdTunnelMaster; //Indy
22464: 454 unit uPSI_IdTunnelSlave; //Indy
22465: 455 unit uPSI_IdRSH; //Indy
22466: 456 unit uPSI_IdRSHServer; //Indy
22467: 457 unit uPSI_Spring_Cryptography_Utils; //Spring4Delphi

```

```

22468: 458 unit uPSI_MapReader; //devC
22469: 459 unit uPSI_LibTar; //devC
22470: 460 unit uPSI_IdStack; //Indy
22471: 461 unit uPSI_IdbBlockCipherIntercept; //Indy
22472: 462 unit uPSI_IdChargenServer; //Indy
22473: 463 unit uPSI_IdFTPServer; //Indy
22474: 464 unit uPSI_IdException; //Indy
22475: 465 unit uPSI_Utexplot; //DMath
22476: 466 unit uPSI_uwinstr; //DMath
22477: 467 unit uPSI_VarRecUtils; //devC
22478: 468 unit uPSI_JvStringListToHtml; //JCL
22479: 469 unit uPSI_JvStringHolder; //JCL
22480: 470 unit uPSI_IdCoder; //Indy
22481: 471 unit uPSI_SynHighlighterDfm; //Synedit
22482: 472 unit uHighlighterProcs; in 471 //Synedit
22483: 473 unit uPSI_LazFileUtils; //LCL
22484: 474 unit uPSI_IDECmdLine; //LCL
22485: 475 unit uPSI_lazMasks; //LCL
22486: 476 unit uPSI_ip_misc; //mX4
22487: 477 unit uPSI_Barcodes; //LCL
22488: 478 unit uPSI_SimpleXML; //LCL
22489: 479 unit uPSI_JclIniFiles; //JCL
22490: 480 unit uPSI_D2XXUnit; {$X-} //FTDI
22491: 481 unit uPSI_JclDateTime; //JCL
22492: 482 unit uPSI_JclEDI; //JCL
22493: 483 unit uPSI_JclMiscel2; //JCL
22494: 484 unit uPSI_JclValidation; //JCL
22495: 485 unit uPSI_JclAnsiStrings; {-PString} //JCL
22496: 486 unit uPSI_SynEditMiscProcs2; //Synedit
22497: 487 unit uPSI_JclStreams; //JCL
22498: 488 unit uPSI_QRCode; //mX4
22499: 489 unit uPSI_BlockSocket; //ExtPascal
22500: 490 unit uPSI_Masks_Utils; //VCL
22501: 491 unit uPSI_synautil; //Synapse!
22502: 492 unit uPSI_JclMath_Class; //JCL RTL
22503: 493 unit ugamdist; //Gamma function //DMath
22504: 494 unit uibeta, ucorrel; //IBeta //DMath
22505: 495 unit uPSI_SRMgr; //mX4
22506: 496 unit uPSI_HotLog; //mX4
22507: 497 unit uPSI_DebugBox; //mX4
22508: 498 unit uPSI_ustrings; //DMath
22509: 499 unit uPSI_uregtest; //DMath
22510: 500 unit uPSI_usimplex; //DMath
22511: 501 unit uPSI_uhyper; //DMath
22512: 502 unit uPSI_IdHL7; //Indy
22513: 503 unit uPSI_IdIPMCastBase; //Indy
22514: 504 unit uPSI_IdIPMCastServer; //Indy
22515: 505 unit uPSI_IdIPMCastClient; //Indy
22516: 506 unit uPSI_unlfit; //nlregression //DMath
22517: 507 unit uPSI_IdRawHeaders; //Indy
22518: 508 unit uPSI_IdRawClient; //Indy
22519: 509 unit uPSI_IdRawFunctions; //Indy
22520: 510 unit uPSI_IdTCPStream; //Indy
22521: 511 unit uPSI_IdSNPP; //Indy
22522: 512 unit uPSI_St2DBarC; //SysTools
22523: 513 unit uPSI_ImageWin; //FTL //VCL
22524: 514 unit uPSI_CustomDrawTreeView; //FTL //VCL
22525: 515 unit uPSI_GraphWin; //FTL //VCL
22526: 516 unit uPSI_actionMain; //FTL //VCL
22527: 517 unit uPSI_StSpawn; //SysTools
22528: 518 unit uPSI_CtlPanel; //VCL
22529: 519 unit uPSI_IdLPR; //Indy
22530: 520 unit uPSI_SockRequestInterpreter; //Indy
22531: 521 unit uPSI_ultimo; //DMath
22532: 522 unit uPSI_ucholesk; //DMath
22533: 523 unit uPSI_SimpleDS; //VCL
22534: 524 unit uPSI_DBXSqlScanner; //VCL
22535: 525 unit uPSI_DBXMetaDataUtil; //VCL
22536: 526 unit uPSI_Chart; //TEE
22537: 527 unit uPSI_TeeProcs; //TEE
22538: 528 unit mXBDEUtils; //mX4
22539: 529 unit uPSI_MDIEdit; //VCL
22540: 530 unit uPSI_CopyPsr; //VCL
22541: 531 unit uPSI_SockApp; //VCL
22542: 532 unit uPSI_AppEvnts; //VCL
22543: 533 unit uPSI_ExtActns; //VCL
22544: 534 unit uPSI_TeEngine; //TEE
22545: 535 unit uPSI_CoolMain; //browser //VCL
22546: 536 unit uPSI_StCRC; //SysTools
22547: 537 unit uPSI_StDecMth2; //SysTools
22548: 538 unit uPSI_frmExportMain; //Synedit
22549: 539 unit uPSI_SynDBEdit; //Synedit
22550: 540 unit uPSI_SynEditWildcardSearch; //Synedit
22551: 541 unit uPSI_BoldComUtils; //BOLD
22552: 542 unit uPSI_BoldIsoDateTime; //BOLD
22553: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
22554: 544 unit uPSI_BoldXMLRequests; //BOLD
22555: 545 unit uPSI_BoldStringList; //BOLD
22556: 546 unit uPSI_BoldFileHandler; //BOLD

```

```

22557: 547 unit uPSI_BoldContainers; //BOLD
22558: 548 unit uPSI_BoldQueryUserDlg; //BOLD
22559: 549 unit uPSI_BoldWinINet; //BOLD
22560: 550 unit uPSI_BoldQueue; //BOLD
22561: 551 unit uPSI_JvPcx; //JCL
22562: 552 unit uPSI_IdWhois; //Indy
22563: 553 unit uPSI_IdWhoisServer; //Indy
22564: 554 unit uPSI_IdGopher; //Indy
22565: 555 unit uPSI_IdDateTimeStamp; //Indy
22566: 556 unit uPSI_IdDayTimeServer; //Indy
22567: 557 unit uPSI_IdDayTimeUDP; //Indy
22568: 558 unit uPSI_IdDayTimeUDPServer; //Indy
22569: 559 unit uPSI_IdDICTServer; //Indy
22570: 560 unit uPSI_IdDiscardServer; //Indy
22571: 561 unit uPSI_IdDiscardUDPServer; //Indy
22572: 562 unit uPSI_IdMappedFTP; //Indy
22573: 563 unit uPSI_IdMappedPortTCP; //Indy
22574: 564 unit uPSI_IdGopherServer; //Indy
22575: 565 unit uPSI_IdQotdServer; //Indy
22576: 566 unit uPSI_JvRgbToHtml; //JCL
22577: 567 unit uPSI_JvRemLog; //JCL
22578: 568 unit uPSI_JvSysComp; //JCL
22579: 569 unit uPSI_JvTMTL; //JCL
22580: 570 unit uPSI_JvWinampAPI; //JCL
22581: 571 unit uPSI_MSysUtils; //mX4
22582: 572 unit uPSI_ESBMaths; //ESB
22583: 573 unit uPSI_ESBMaths2; //ESB
22584: 574 unit uPSI_uLkJSON; //Lk
22585: 575 unit uPSI_ZURLi //Zeos //Zeos
22586: 576 unit uPSI_ZSysUtils; //Zeos
22587: 577 unit unaUtils internals //UNA
22588: 578 unit uPSI_ZMatchPattern; //Zeos
22589: 579 unit uPSI_ZClasses; //Zeos
22590: 580 unit uPSI_ZCollections; //Zeos
22591: 581 unit uPSI_ZEncoding; //Zeos
22592: 582 unit uPSI_IdRawBase; //Indy
22593: 583 unit uPSI_IdNTLM; //Indy
22594: 584 unit uPSI_IdNNTP; //Indy
22595: 585 unit uPSI_usniffer; //PortScanForm
22596: 586 unit uPSI_IdCoderMIME; //Indy
22597: 587 unit uPSI_IdCoderUUE; //Indy
22598: 588 unit uPSI_IdCoderXXE; //Indy
22599: 589 unit uPSI_IdCoder3to4; //Indy
22600: 590 unit uPSI_IdCookie; //Indy
22601: 591 unit uPSI_IdCookieManager; //Indy
22602: 592 unit uPSI_WDOSocketUtils; //WDOS
22603: 593 unit uPSI_WDOSPlcUtils; //WDOS
22604: 594 unit uPSI_WDOSPorts; //WDOS
22605: 595 unit uPSI_WDOSResolvers; //WDOS
22606: 596 unit uPSI_WDOSTimers; //WDOS
22607: 597 unit uPSI_WDOSPlcs; //WDOS
22608: 598 unit uPSI_WDOSpneumatics; //WDOS
22609: 599 unit uPSI_IdFingerServer; //Indy
22610: 600 unit uPSI_IdDNSResolver; //Indy
22611: 601 unit uPSI_IdHTTPWebBrokerBridge; //Indy
22612: 602 unit uPSI_IdIntercept; //Indy
22613: 603 unit uPSI_IdIPMCastBase; //Indy
22614: 604 unit uPSI_IdLogBase; //Indy
22615: 605 unit uPSI_IdIOHandlerStream; //Indy
22616: 606 unit uPSI_IdMappedPortUDP; //Indy
22617: 607 unit uPSI_IdQOTDUDPServer; //Indy
22618: 608 unit uPSI_IdQOTDUDP; //Indy
22619: 609 unit uPSI_IdSysLog; //Indy
22620: 610 unit uPSI_IdSysLogServer; //Indy
22621: 611 unit uPSI_IdSysLogMessage; //Indy
22622: 612 unit uPSI_IdTimeServer; //Indy
22623: 613 unit uPSI_IdTimeUDP; //Indy
22624: 614 unit uPSI_IdTimeUDPServer; //Indy
22625: 615 unit uPSI_IdUserAccounts; //Indy
22626: 616 unit uPSI_TextUtils; //mX4
22627: 617 unit uPSI_MandelbrotEngine; //mX4
22628: 618 unit uPSI_delphi_arduino_Unit1; //mX4
22629: 619 unit uPSI_DTDSchema2; //mX4
22630: 620 unit uPSI_fplotMain; //DMath
22631: 621 unit uPSI_FindFileIter; //mX4
22632: 622 unit uPSI_PppState; (JclStrHashMap) //PPP
22633: 623 unit uPSI_PppParser; //PPP
22634: 624 unit uPSI_PppLexer; //PPP
22635: 625 unit uPSI_PCharUtils; //PPP
22636: 626 unit uPSI_uJSON; //WU
22637: 627 unit uPSI_JclStrHashMap; //JCL
22638: 628 unit uPSI_JclHookExcept; //JCL
22639: 629 unit uPSI_EncdDecd; //VCL
22640: 630 unit uPSI_SockAppReg; //VCL
22641: 631 unit uPSI_PJFileHandle; //PJ
22642: 632 unit uPSI_PJEnvVars; //PJ
22643: 633 unit uPSI_PJPipe; //PJ
22644: 634 unit uPSI_PJPipeFilters; //PJ
22645: 635 unit uPSI_PJConsoleApp; //PJ

```

```

22646: 636 unit uPSI_UConsoleAppEx; //PJ
22647: 637 unit uPSI_DbxSocketChannelNative; //VCL
22648: 638 unit uPSI_DbxDataGenerator; //VCL
22649: 639 unit uPSI_DBXClient; //VCL
22650: 640 unit uPSI_IdLogEvent; //Indy
22651: 641 unit uPSI_Reversi; //mx4
22652: 642 unit uPSI_Geometry; //mx4
22653: 643 unit uPSI_IdSMTPServer; //Indy
22654: 644 unit uPSI_Textures; //mx4
22655: 645 unit uPSI_IBX; //VCL
22656: 646 unit uPSI_IWDBCommon; //VCL
22657: 647 unit uPSI_SortGrid; //mx4
22658: 648 unit uPSI_IB; //VCL
22659: 649 unit uPSI_IBScript; //VCL
22660: 650 unit uPSI_JvCSVBaseControls; //JCL
22661: 651 unit uPSI_Jvg3DColors; //JCL
22662: 652 unit uPSI_JvHLEditor; //beat //JCL
22663: 653 unit uPSI_JvShellHook; //JCL
22664: 654 unit uPSI_DBCommon2; //VCL
22665: 655 unit uPSI_JvSHFileOperation; //JCL
22666: 656 unit uPSI_uFileexport; //mx4
22667: 657 unit uPSI_JvDialogs; //JCL
22668: 658 unit uPSI_JvDBTreeView; //JCL
22669: 659 unit uPSI_JvDBUltimGrid; //JCL
22670: 660 unit uPSI_JvDBQueryParamsForm; //JCL
22671: 661 unit uPSI_JvExControls; //JCL
22672: 662 unit uPSI_JvBDEMemTable; //JCL
22673: 663 unit uPSI_JvCommStatus; //JCL
22674: 664 unit uPSI_JvMailSlots2; //JCL
22675: 665 unit uPSI_JvgWinMask; //JCL
22676: 666 unit uPSI_StEclipse; //SysTools
22677: 667 unit uPSI_StMime; //SysTools
22678: 668 unit uPSI_StList; //SysTools
22679: 669 unit uPSI_StMerge; //SysTools
22680: 670 unit uPSI_StStrs; //SysTools
22681: 671 unit uPSI_StTree; //SysTools
22682: 672 unit uPSI_StVArr; //SysTools
22683: 673 unit uPSI_StRegIni; //SysTools
22684: 674 unit uPSI_urkf; //DMath
22685: 675 unit uPSI_usvd; //DMath
22686: 676 unit uPSI_DepWalkUtils; //JCL
22687: 677 unit uPSI_OptionsFrm; //JCL
22688: 678 unit yuvconverts; //mx4
22689: 679 uPSI_JvPropAutoSave; //JCL
22690: 680 uPSI_AclAPI; //alcinoe
22691: 681 uPSI_AviCap; //alcinoe
22692: 682 uPSI_ALAVLBinaryTree; //alcinoe
22693: 683 uPSI_ALFcMisc; //alcinoe
22694: 684 uPSI_ALStringList; //alcinoe
22695: 685 uPSI_ALQuickSortList; //alcinoe
22696: 686 uPSI_ALStaticText; //alcinoe
22697: 687 uPSI_ALJSONDoc; //alcinoe
22698: 688 uPSI_ALGSMComm; //alcinoe
22699: 689 uPSI_ALWindows; //alcinoe
22700: 690 uPSI_ALMultiPartFormDataParser; //alcinoe
22701: 691 uPSI_ALHttpCommon; //alcinoe
22702: 692 uPSI_ALWebSpider; //alcinoe
22703: 693 uPSI_ALHttpClient; //alcinoe
22704: 694 uPSI_ALFcHTML; //alcinoe
22705: 695 uPSI_ALFTPClient; //alcinoe
22706: 696 uPSI_ALInternetMessageCommon; //alcinoe
22707: 697 uPSI_ALWininetHttpClient; //alcinoe
22708: 698 uPSI_ALWinInetFTPCClient; //alcinoe
22709: 699 uPSI_ALWinHttpWrapper; //alcinoe
22710: 700 uPSI_ALWinHttpClient; //alcinoe
22711: 701 uPSI_ALFcWinSock; //alcinoe
22712: 702 uPSI_ALFcSQL; //alcinoe
22713: 703 uPSI_ALFcCGI; //alcinoe
22714: 704 uPSI_ALFcExecute; //alcinoe
22715: 705 uPSI_ALFcFile; //alcinoe
22716: 706 uPSI_ALFcMimeType; //alcinoe
22717: 707 uPSI_ALPhpRunner; //alcinoe
22718: 708 uPSI_ALGraphic; //alcinoe
22719: 709 uPSI_ALIniFiles; //alcinoe
22720: 710 uPSI_ALMemCachedClient; //alcinoe
22721: 711 unit uPSI_MyGrids; //mx4
22722: 712 uPSI_ALMultiPartMixedParser //alcinoe
22723: 713 uPSI_ALSMTPClient //alcinoe
22724: 714 uPSI_ALNNTPClient; //alcinoe
22725: 715 uPSI_ALHintBalloon; //alcinoe
22726: 716 unit uPSI_ALXmlDoc; //alcinoe
22727: 717 unit uPSI_IPCThrd; //VCL
22728: 718 unit uPSI_MonForm; //VCL
22729: 719 unit uPSI_TeCanvas; //Orpheus
22730: 720 unit uPSI_Ovcmisc; //Orpheus
22731: 721 unit uPSI_ovcfiler; //Orpheus
22732: 722 unit uPSI_ovcstate; //Orpheus
22733: 723 unit uPSI_ovccoco; //Orpheus
22734: 724 unit uPSI_ovcrvexp; //Orpheus

```

```

22735: 725 unit uPSI_OvcFormatSettings;           //Orpheus
22736: 726 unit uPSI_OvcUtils;                   //Orpheus
22737: 727 unit uPSI_ovcstore;                  //Orpheus
22738: 728 unit uPSI_ocvstr;                   //Orpheus
22739: 729 unit uPSI_ovcmru;                  //Orpheus
22740: 730 unit uPSI_ovccmd;                  //Orpheus
22741: 731 unit uPSI_ovctimer;                //Orpheus
22742: 732 unit uPSI_ovcintl;                 //Orpheus
22743: 733 uPSI_AfCircularBuffer;             //AsyncFree
22744: 734 uPSI_AfUtils;                     //AsyncFree
22745: 735 uPSI_AfSafeSync;                  //AsyncFree
22746: 736 uPSI_AfComPortCore;              //AsyncFree
22747: 737 uPSI_AfComPort;                  //AsyncFree
22748: 738 uPSI_AfPortControls;            //AsyncFree
22749: 739 uPSI_AfViewers;                  //AsyncFree
22750: 740 uPSI_AfDataDispatcher;          //AsyncFree
22751: 741 uPSI_AfDataTerminal;            //AsyncFree
22752: 742 uPSI_SimplePortMain;           //AsyncFree
22753: 743 unit uPSI_ovcclock;            //Orpheus
22754: 744 unit uPSI_o32intlst;           //Orpheus
22755: 745 unit uPSI_o32ledlabel;         //Orpheus
22756: 746 unit uPSI_AlMySqlClient;       //alcinoe
22757: 747 unit uPSI_ALFBXClient;        //alcinoe
22758: 748 unit uPSI_ALFcnsQL;          //alcinoe
22759: 749 unit uPSI_AsyncTimer;         //mX4
22760: 750 unit uPSI_ApplicationFileIO;   //mX4
22761: 751 unit uPSI_PsAPI;              //VCLé
22762: 752 uPSI_ovcuser;                 //Orpheus
22763: 753 uPSI_ovcurl;                 //Orpheus
22764: 754 uPSI_ovcvlb;                 //Orpheus
22765: 755 uPSI_ovccolor;              //Orpheus
22766: 756 uPSI_ALFBXLib;              //alcinoe
22767: 757 uPSI_ovcmeter;              //Orpheus
22768: 758 uPSI_ovcpeakm;             //Orpheus
22769: 759 uPSI_O32BGSty;              //Orpheus
22770: 760 uPSI_ovcBidi;                //Orpheus
22771: 761 uPSI_ovctcarry;            //Orpheus
22772: 762 uPSI_DXPUtilities;         //mX4
22773: 763 uPSI_ALMultiPartBaseParser; //alcinoe
22774: 764 uPSI_ALMultiPartAlternativeParser; //alcinoe
22775: 765 uPSI_ALPOP3Client;         //alcinoe
22776: 766 uPSI_SmallUtils;            //mX4
22777: 767 uPSI_MakeApp;              //mX4
22778: 768 uPSI_O32MouseMon;          //Orpheus
22779: 769 uPSI_OvcCache;             //Orpheus
22780: 770 uPSI_ovccalc;              //Orpheus
22781: 771 uPSI_Joystick;            //OpenGL
22782: 772 uPSI_ScreenSaver;          //OpenGL
22783: 773 uPSI_XCollection;         //OpenGL
22784: 774 uPSI_Polynomials;          //OpenGL
22785: 775 uPSI_PersistentClasses;    //9.86
22786: 776 uPSI_VectorLists;          //OpenGL
22787: 777 uPSI_XOpenGL;              //OpenGL
22788: 778 uPSI_MeshUtils;            //OpenGL
22789: 779 unit uPSI_JclSysUtils;    //JCL
22790: 780 unit uPSI_JclBorlandTools; //JCL
22791: 781 unit Jcl FileUtils_max;   //JCL
22792: 782 uPSI_AfDataControls;      //AsyncFree
22793: 783 uPSI_GLSilhouette;        //OpenGL
22794: 784 uPSI_JclSysUtils_class;  //JCL
22795: 785 uPSI_Jcl FileUtils_class; //JCL
22796: 786 uPSI_FileUtil;            //JCL
22797: 787 uPSI_changefind;          //mX4
22798: 788 uPSI_CmdIntf;             //mX4
22799: 789 uPSI_fservice;            //mX4
22800: 790 uPSI_Keyboard;            //OpenGL
22801: 791 uPSI_VRMLParser;          //OpenGL
22802: 792 uPSI_GLFileVRML;         //OpenGL
22803: 793 uPSI_Octree;              //OpenGL
22804: 794 uPSI_GLPolyhedron;        //OpenGL
22805: 795 uPSI_GLCrossPlatform;    //OpenGL
22806: 796 uPSI_GLParticles;        //OpenGL
22807: 797 uPSI_GLNavigator;         //OpenGL
22808: 798 uPSI_GLStarRecord;       //OpenGL
22809: 799 uPSI_GLTextureCombiners; //OpenGL
22810: 800 uPSI_GLCanvas;            //OpenGL
22811: 801 uPSI_GeometryBB;          //OpenGL
22812: 802 uPSI_GeometryCoordinates; //OpenGL
22813: 803 uPSI_VectorGeometry;     //OpenGL
22814: 804 uPSI_BumpMapping;         //OpenGL
22815: 805 uPSI_TGA;                //OpenGL
22816: 806 uPSI_GLVectorFileObjects; //OpenGL
22817: 807 uPSI_IMM;                //VCL
22818: 808 uPSI_CategoryButtons;    //VCL
22819: 809 uPSI_ButtonGroup;        //VCL
22820: 810 uPSI_DbExcept;            //VCL
22821: 811 uPSI_AxCtrls;             //VCL
22822: 812 uPSI_GL_actorUnit1;     //OpenGL
22823: 813 uPSI_StdVCL;             //VCL

```

```

22824: 814 unit CurvesAndSurfaces;                                //OpenGL
22825: 815 uPSI_DataAwareMain;                                 //AsyncFree
22826: 816 uPSI_TabNotBk;                                    //VCL
22827: 817 uPSI_udwsfiler;                                 //mX4
22828: 818 uPSI_synaip;                                   //Synapse!
22829: 819 uPSI_synacode;                                //Synapse
22830: 820 uPSI_synachar;                                //Synapse
22831: 821 uPSI_synamisc;                               //Synapse
22832: 822 uPSI_synaser;                                //Synapse
22833: 823 uPSI_synaicnv;                               //Synapse
22834: 824 uPSI_tlntsend;                                //Synapse
22835: 825 uPSI_pingsend;                               //Synapse
22836: 826 uPSI_blksock;                                //Synapse
22837: 827 uPSI_asnlutil;                               //Synapse
22838: 828 uPSI_dnssend;                                //Synapse
22839: 829 uPSI_clamsend;                               //Synapse
22840: 830 uPSI_ldapsend;                               //Synapse
22841: 831 uPSI_mimemess;                               //Synapse
22842: 832 uPSI_slogsend;                               //Synapse
22843: 833 uPSI_mimepart;                               //Synapse
22844: 834 uPSI_mimeinln;                               //Synapse
22845: 835 uPSI_ftpsend;                                //Synapse
22846: 836 uPSI_ftptsend;                               //Synapse
22847: 837 uPSI_httpsend;                               //Synapse
22848: 838 uPSI_sntpsend;                               //Synapse
22849: 839 uPSI_smtpsend;                               //Synapse
22850: 840 uPSI_snmpsend;                               //Synapse
22851: 841 uPSI_imapsend;                               //Synapse
22852: 842 uPSI_pop3send;                               //Synapse
22853: 843 uPSI_nntpsend;                               //Synapse
22854: 844 uPSI_ssl_cryptlib;                            //Synapse
22855: 845 uPSI_ssl_openssl;                            //Synapse
22856: 846 uPSI_synhttp_daemon;                           //Synapse
22857: 847 uPSI_NetWork;                                //mX4
22858: 848 uPSI_PingThread;                             //Synapse
22859: 849 uPSI_JvThreadTimer;                            //JCL
22860: 850 unit uPSI_wwSystem;                            //InfoPower
22861: 851 unit uPSI_IdComponent;                           //Indy
22862: 852 unit uPSI_IdIOHandlerThrottle;                //Indy
22863: 853 unit uPSI_Themes;                                //VCL
22864: 854 unit uPSI_StdStyleActnCtrls;                  //VCL
22865: 855 unit uPSI_UDDIHelper;                            //VCL
22866: 856 unit uPSI_IdIMAP4Server;                      //Indy
22867: 857 uPSI_VariantSymbolTable;                         //VCL //3.9.9.92
22868: 858 uPSI_udf_glob;                                //mX4
22869: 859 uPSI_TabGrid;                                //VCL
22870: 860 uPSI_JsDBTreeView;                            //mX4
22871: 861 uPSI_JsSendMail;                               //mX4
22872: 862 uPSI_dbTvRecordList;                            //mX4
22873: 863 uPSI_TreeWEx;                                //mX4
22874: 864 uPSI_ECDataLink;                            //mX4
22875: 865 uPSI_dbTree;                                //mX4
22876: 866 uPSI_dbTreeCBox;                            //mX4
22877: 867 unit uPSI_Debug; //TfrmDebug;                  //mX4
22878: 868 uPSI_TimeFncs;                                //mX4
22879: 869 uPSI_FileIntf;                                //VCL
22880: 870 uPSI_SockTransport;                            //RTL
22881: 871 unit uPSI_WinInet;                            //RTL
22882: 872 unit uPSI_WWSTR;                                //mX4
22883: 873 uPSI_DBLookup;                                //VCL
22884: 874 uPSI_Hotspot;                                //mX4
22885: 875 uPSI_HList; //History List;                  //mX4
22886: 876 unit uPSI_DrTable;                            //VCL
22887: 877 uPSI_TConnect;                                //VCL
22888: 878 uPSI_DataBkr;                                //VCL
22889: 879 uPSI_HTTPIntr;                                //VCL
22890: 880 unit uPSI_Mathbox;                            //mX4
22891: 881 uPSI_cyIndy;                                //cY
22892: 882 uPSI_cySysUtils;                            //cY
22893: 883 uPSI_cyWinUtils;                            //cY
22894: 884 uPSI_cyStrUtils;                            //cY
22895: 885 uPSI_cyObjUtils;                            //cY
22896: 886 uPSI_cyDateUtils;                            //cY
22897: 887 uPSI_cyBDE;                                //cY
22898: 888 uPSI_cyClasses;                            //cY
22899: 889 uPSI_cyGraphics; //3.9.9.94_2;           //cY
22900: 890 unit uPSI_cyTypes;                            //cY
22901: 891 uPSI_JvDateTimePicker;                      //JCL
22902: 892 uPSI_JvCreateProcess;                          //JCL
22903: 893 uPSI_JvEasterEgg;                            //JCL
22904: 894 uPSI_WinSvc; //3.9.9.94_3;             //VCL
22905: 895 uPSI_SvcMgr;                                //VCL
22906: 896 unit uPSI_JvPickDate;                          //JCL
22907: 897 unit uPSI_JvNotify;                            //JCL
22908: 898 uPSI_JvStrHlder;                            //JCL
22909: 899 unit uPSI_JclNTFS2;                            //JCL
22910: 900 uPSI_Jcl8087 //math coprocessor;           //JCL
22911: 901 uPSI_JvAddPrinter;                            //JCL
22912: 902 uPSI_JvCabFile;                                //JCL

```

```

22913: 903 uPSI_JvDataEmbedded; //JCL
22914: 904 unit uPSI_U_HexView; //mX4
22915: 905 uPSI_UWavein4; //mX4
22916: 906 uPSI_AMixer; //mX4
22917: 907 uPSI_JvaScrollText; //mX4
22918: 908 uPSI_JvArrow; //mX4
22919: 909 unit uPSI.UrlMon; //mX4
22920: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas' //mX4
22921: 911 unit uPSI_U_Oscilloscope4; //TOScfMain; //DFF
22922: 912 unit uPSI_DFFUtils; //DFF
22923: 913 unit uPSI_MathsLib; //DFF
22924: 914 uPSI_UIntList; //DFF
22925: 915 uPSI_UGetParents; //DFF
22926: 916 unit uPSI_UGeometry; //DFF
22927: 917 unit uPSI_UAstronomy; //DFF
22928: 918 unit uPSI_UCardComponentV2; //DFF
22929: 919 unit uPSI_UTGraphSearch; //DFF
22930: 920 unit uPSI_UParser10; //DFF
22931: 921 unit uPSI_cyIEUtils; //cY
22932: 922 unit uPSI_UcomboV2; //DFF
22933: 923 uPSI_cyBaseComm; //cY
22934: 924 uPSI_cyAppInstances; //cY
22935: 925 uPSI_cyAttract; //cY
22936: 926 uPSI_cyDERUtils; //cY
22937: 927 unit uPSI_cyDocER; //cY
22938: 928 unit uPSI_ODBC; //mX
22939: 929 unit uPSI_AssocExec; //mX
22940: 930 uPSI_cyBaseCommRoomConnector; //cY
22941: 931 uPSI_cyCommRoomConnector; //cY
22942: 932 uPSI_cyCommunicate; //cY
22943: 933 uPSI_cyImage; //cY
22944: 934 uPSI_cyBaseContainer; //cY
22945: 935 uPSI_cyModalContainer; //cY
22946: 936 uPSI_cyFlyingContainer; //cY
22947: 937 uPSI_RegStr; //VCL
22948: 938 uPSI_HtmlHelpViewer; //VCL
22949: 939 unit uPSI_cyIniform; //cY
22950: 940 unit uPSI_cyVirtualGrid; //cY
22951: 941 uPSI_Profiler; //DA
22952: 942 uPSI_BackgroundWorker; //DA
22953: 943 uPSI_Waveplay; //DA
22954: 944 uPSI_WaveTimer; //DA
22955: 945 uPSI_WaveUtils; //DA
22956: 946 uPSI_NamedPipes; //TB
22957: 947 uPSI_NamedPipeServer; //TB
22958: 948 unit uPSI_process; //TB
22959: 949 unit uPSI_DPUtills; //TB
22960: 950 unit uPSI_CommonTools; //TB
22961: 951 uPSI_DataSendToWeb; //TB
22962: 952 uPSI_StarCalc; //TB
22963: 953 uPSI_D2_XPVistaHelperU //TB
22964: 954 unit uPSI_NetTools; //TB
22965: 955 unit uPSI_Pipes; //TB
22966: 956 uPSI_ProcessUnit; //mX
22967: 957 uPSI_adGSM; //mX
22968: 958 unit uPSI_BetterADODataSet; //mX
22969: 959 unit uPSI_AdSelCom; //FTT //mX
22970: 960 unit unit uPSI_dwsXPlatform; //DWS //mX Turbo Power
22971: 961 uPSI_AdSocket; //mX
22972: 962 uPSI_AdPacket; //mX
22973: 963 uPSI_AdPort; //mX
22974: 964 uPSI_PathFunc; //Inno
22975: 965 uPSI_CmnFunc; //Inno
22976: 966 uPSI_CmnFunc2; //Inno Setup //Inno
22977: 967 unit uPSI_BitmapImage; //mX4
22978: 968 unit uPSI_ImageGrabber; //mX4
22979: 969 uPSI_SecurityFunc; //Inno
22980: 970 uPSI_RedirFunc; //Inno
22981: 971 uPSI_FIFO, (MemoryStream) //mX4
22982: 972 uPSI_Int64Bm; //Inno
22983: 973 unit uPSI_InstFunc; //Inno
22984: 974 unit uPSI_LibFusion; //Inno
22985: 975 uPSI_SimpleExpression; //Inno
22986: 976 uPSI_unitResourceDetails; //XN
22987: 977 uPSI_unitResFile; //XN
22988: 978 unit uPSI_simpleComport; //mX4
22989: 979 unit uPSI_AfViewershelfers; //Async
22990: 980 unit uPSI_Console; //mX4
22991: 981 unit uPSI_AnalogMeter; //TB
22992: 982 unit uPSI_XPrinter; //TB
22993: 983 unit uPSI_IniFiles; //VCL
22994: 984 unit uPSI_lazIniFiles; //FP
22995: 985 uPSI_testutils; //FP
22996: 986 uPSI_ToolsUnit; (DBTests) //FP
22997: 987 uPSI_fpcunit; //FP
22998: 988 uPSI_testdecorator; //FP
22999: 989 unit uPSI_fpcunittests; //FP
23000: 990 unit uPSI_cTCPBuffer; //Fundamentals 4
23001: 991 unit uPSI_Glut, //Open GL

```

```

23002: 992 uPSI_LEDBitmaps, //mX4
23003: 993 uPSI_FileClass, //Inno
23004: 994 uPSI_FileUtilsClass, //mX4
23005: 995 uPSI_ComPortInterface; //Kit //mX4
23006: 996 unit uPSI_SwitchLed; //mX4
23007: 997 unit uPSI_cyDmmCanvas; //cY
23008: 998 uPSI_uColorFunctions; //DFF
23009: 999 uPSI_uSettings; //DFF
23010: 1000 uPSI_cyDebug.pas //cY
23011: 1001 uPSI_cyColorMatrix; //cY
23012: 1002 unit uPSI_cyCopyFiles; //cY
23013: 1003 unit uPSI_cySearchFiles; //cY
23014: 1004 unit uPSI_cyBaseMeasure; //cY
23015: 1005 unit uPSI_PJStreams; //DD
23016: 1006 unit uPSI_cyRunTimeResize; //cY
23017: 1007 unit uPSI_jcontrolutils; //cY
23018: 1008 unit uPSI_kcMapView; (+GEONames) //kc
23019: 1009 unit uPSI_kcMapViewDESynapse; //kc
23020: 1010 unit uPSI_cparserutils; (+GIS_SysUtils) //kc
23021: 1011 unit uPSI_LedNumber; //TurboPower
23022: 1012 unit uPSI_StStrL; //SysTools
23023: 1013 unit uPSI_IndGnouMeter; //LAZ
23024: 1014 unit uPSI_Sensors; //LAZ
23025: 1015 unit uPSI_pwmain; //cgi of powtils //Pow
23026: 1016 unit uPSI_HTMLUtil; //Pow
23027: 1017 unit uPSI_synwrap1; //httpsend //Pow
23028: 1018 unit StreamWrap1 //Pow
23029: 1019 unit uPSI_pwmain; //Pow
23030: 1020 unit pwtypes //Pow
23031: 1021 uPSI_W32VersionInfo //LAZ
23032: 1022 unit uPSI_IpAnim; //LAZ
23033: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
23034: 1024 unit uPSI_LrtPoTools; //LAZ
23035: 1025 unit uPSI_Laz_DOM; //LAZ
23036: 1026 unit uPSI_hhAvComp; //LAZ
23037: 1027 unit uPSI_GPS2; //mX4
23038: 1028 unit uPSI_GPS; //mX4
23039: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo //mX4
23040: 1030 unit uPSI_NMEA; // GPS //mX4
23041: 1031 unit uPSI_ScreenThreeeDLab; //mX4
23042: 1032 unit uPSI_Spin; //VCL
23043: 1033 unit uPSI_DynaZip; //mX4
23044: 1034 unit uPSI_clockExpert; //TB
23045: 1035 unit debugLn //mX4
23046: 1036 uPSI_SortUtils; //Jcl
23047: 1037 uPSI_BitmapConversion; //Jcl
23048: 1038 unit uPSI_JclTD32; //Jcl
23049: 1039 unit uPSI_ZDbcUtils; //Zeos
23050: 1040 unit uPSI_ZScriptParser; //Zeos
23051: 1041 uPSI_JvIni; //JCL
23052: 1042 uPSI_JvFtpGrabber; //JCL
23053: 1043 unit uPSI_NeuralNetwork; //OCL
23054: 1044 unit uPSI_StExpr; //SysTools
23055: 1045 unit uPSI_GR32_Geometry; //GR32
23056: 1046 unit uPSI_GR32_Containers; //GR32
23057: 1047 unit uPSI_GR32_Backends_VCL; //GR32
23058: 1048 unit uPSI_StSaturn; //Venus+Mercury+Mars++ //SysTools
23059: 1049 unit uPSI_JclParseUses; //JCL
23060: 1050 unit uPSI_JvFinalize; //JCL
23061: 1051 unit uPSI_panUnit1; //GLScene
23062: 1052 unit uPSI_DD83u1; //Arduino Tester //mX4
23063: 1053 unit uPSI_BigIni //Hinzen
23064: 1054 unit uPSI_ShellCtrls; //VCL
23065: 1055 unit uPSI_fmath; //FMath
23066: 1056 unit uPSI_fComp; //FMath
23067: 1057 unit uPSI_HighResTimer; //Lauer
23068: 1058 unit uconvMain; (Unit Converter) //PS
23069: 1059 uPSI_uconvMain; //PS
23070: 1060 unit uPSI_ParserUtils; //PS
23071: 1061 unit uPSI_UPSUtils; //PS
23072: 1062 unit uPSI_ParserU; //PS
23073: 1063 unit uPSI_TypeInfo; //VCL
23074: 1064 unit uPSI_ServiceMgr; //mX
23075: 1065 unit uPSI_UDict; //DFF
23076: 1066 unit uPSI_ubigFloatV3; //DFF
23077: 1067 unit uPSI_UBigIntsV4; //DFF
23078: 1068 unit uPSI_ServiceMgr2; //mX
23079: 1069 unit uPSI_UP10Build; //PS
23080: 1070 unit uPSI_UParser10; //PS
23081: 1071 unit uPSI_IdModBusServer; //MB
23082: 1072 unit uPSI_IdModBusClient; //MB
23083: 1073 unit uPSI_ColorGrd; //VCL
23084: 1074 unit uPSI_DirOutln; //VCL
23085: 1075 unit uPSI_Gauges; //VCL
23086: 1076 unit uPSI_CustomizeDlg; //VCL
23087: 1077 unit uPSI_ActnMan; //VCL
23088: 1078 unit uPSI_CollPanl; //VCL
23089: 1079 unit uPSI_Calendar2; //VCL
23090: 1080 unit uPSI_IBCtrls; //VCL

```

```

23091: 1081 unit uPSI_IdStackWindows;                                //Indy
23092: 1082 unit uPSI_CTSVendorUtils;                               //DBX
23093: 1083 unit uPSI_VendorTestFramework;                            //DBX
23094: 1084 unit uPSI_TInterval;                                    //mX4
23095: 1085 unit uPSI_JvAnimate;                                     //JCL
23096: 1086 unit uPSI_DBXCharDecoder;                                //DBX
23097: 1087 unit uPSI_JvDBLists;                                    //JCL
23098: 1088 unit uPSI_JvFileInfo;                                   //JCL
23099: 1089 unit uPSI_SOAPConn;                                    //VCL
23100: 1090 unit uPSI_SOAPLinked;                                 //VCL
23101: 1091 unit uPSI_XSBuiltIns;                                  //VCL
23102: 1092 unit uPSI_JvgDigits;                                 //JCL
23103: 1093 unit uPSI_JvDesignUtils;    1094 unit uPSI_JvgCrossTable;
23104: 1095 unit uPSI_JvgReport;        1096 unit uPSI_JvDBRichEdit;
23105: 1097 unit uPSI_JvWinHelp;       1098 unit uPSI_WaveConverter;
23106: 1099 unit uPSI_ACMDConvertor;   1100 unit XSBuiltIns_Routines
23107:
23108:
23109: http://www.slideshare.net/maxkleiner1/codereview-topics
23110: //////////////////////////////////////////////////////////////////
23111: //Form Template Library FTL
23112: //////////////////////////////////////////////////////////////////
23113:
23114:     FTL For Form Building Lib out of the Script, eg. 399_form_templates.txt
23115: 045 unit uPSI_VListView;                                         TFormListView;
23116: 263 unit uPSI_JvProfiler32;                                    TProfReport
23117: 270 unit uPSI_ImgList;                                         TCustomImageList
23118: 278 unit uPSI_JvImageWindow;                                TJvImageWindow
23119: 317 unit uPSI_JvParserForm;                                TJvHTMLParserForm
23120: 497 unit uPSI_DebugBox;                                      TDebugBox
23121: 513 unit uPSI_ImageWin;                                     TImageForm, TImageForm2
23122: 514 unit uPSI_CustomDrawTreeView;                           TCustomDrawForm
23123: 515 unit uPSI_GraphWin;                                    TGraphWinForm
23124: 516 unit uPSI_actionMain;                                 TActionForm
23125: 518 unit uPSI_CtlPanel;                                    TAppletApplication
23126: 529 unit uPSI_MDIEdit;                                     TEditForm //RichEditApp
23127: 535 unit uPSI_CoolMain; {browser}                         TWebMainForm
23128: 538 unit uPSI_frmExportMain;                             TSynexportForm
23129: 585 unit uPSI_usniffer; { //PortScanForm}
23130: 600 unit uPSI_ThreadForm;                                 TThreadSortForm;
23131: 618 unit uPSI_delphi_arduino_Unit1;                      TLEDForm
23132: 620 unit uPSI_fplotMain;                                 TfplotForm1
23133: 660 unit uPSI_JvDBQueryParamsForm;                      TJvQueryParamsDialog
23134: 677 unit uPSI_OptionsFrm;                                TfrmOptions;
23135: 718 unit uPSI_MonForm;                                    TMonitorForm
23136: 742 unit uPSI_SimplePortMain;                           TPortForm1
23137: 770 unit uPSI_ovccalc;                                    TOvcCalculator //widget
23138: 810 unit uPSI_DbExcept;                                 TDbEngineErrorDlg
23139: 812 unit uPSI_GL_actorUnit1;                           TglActorForm1 //OpenGL Robot
23140: 846 unit uPSI_synhttp_daemon;                          TTCPHttpDaemon, TTCPHttpThrd, TPingThread
23141: 867 unit uPSI_Debug;                                     TfrmDebug
23142: 901 unit uPSI_JvAddPrinter;                            TJvAddPrinter //JCL
23143: 904 unit uPSI_U_HexView;                                THexForm2
23144: 911 unit uPSI_U_Oscilloscope4; {OscfrmMain}
23145: 959 unit uPSI_AdSelCom;                                TComSelectForm
23146: 1029 unit uPSI_GPSUDemo;                               TFDemo
23147: 1031 unit uPSI_ScreenThreeDLab;                        TFormLab3D
23148: 1051 unit uPSI_panUnit1;                                TPanForm1 //GLScene
23149: 1052 unit uPSI_DD83ul; {Arduino Tester Frm}
23150: 1059 unit uPSI_uconvMain;                               TfconvMain //PS
23151: 1076 unit uPSI_CustomizeDlg;                           TCustomizeDlg / TJvAddPrinterDialog;
23152:
23153: FormTemplates with <Ctrl> J
23154:
23155: [myformtemplate | formtemplate statement | Borland.EditOptions.Pascal] //with
23156: [myFastForm | class declaration | Borland.EditOptions.Pascal] //Dialog Form
23157: [myForm | class declaration | Borland.EditOptions.Pascal] //with Events
23158: [aForm | class declaration | Borland.EditOptions.Pascal] //single Form
23159:
23160:
23161: procedure SIRegister_JvDesignUtils(CL: TPSPascalCompiler);
23162: begin
23163:   Function DesignClientToParent( const APoint : TPoint; AControl, AParent : TControl ) : TPoint';
23164:   Function DesignMin( AA, AB : Integer ) : Integer';
23165:   Function DesignMax( AA, AB : Integer ) : Integer';
23166:   Function DesignRectWidth( const ARect : TRect ) : Integer';
23167:   Function DesignRectHeight( const ARect : TRect ) : Integer';
23168:   Function DesignValidateRect( const ARect : TRect ) : TRect';
23169:   Function DesignNameIsUnique( AOwner : TComponent; const AName : string ) : Boolean';
23170:   Function DesignUniqueName( AOwner : TComponent; const AClassName : string ) : string');
23171:   Procedure DesignPaintRubberbandRect( AContainer : TWinControl; ARect : TRect; APenStyle : TPenStyle );
23172:   Procedure DesignPaintGrid( ACanvas : TCanvas; const ARect : TRect; ABackColor : TColor; AGridColor : TColor; ADivPixels : Integer );
23173:   Procedure DesignPaintRules( ACanvas : TCanvas; const ARect:TRect; ADivPixels:Integer; ASubDivs: Boolean );
23174:   Procedure DesignSaveComponentToStream( AComp : TComponent; AStream : TStream );
23175:   Function DesignLoadComponentFromStream(AComp:TComponent;AStream:TStream; AOnError:TReaderError):TComponent;
23176:   Procedure DesignSaveComponentToFile( AComp : TComponent; const AFileName : string );
23177:   Procedure DesignLoadComponentFromFile(AComp:TComponent; const AFileName: string; AOnError:TReaderError );

```

```

23178: end;
23179:
23180: ex.:with TEditForm.create(self) do begin
23181:   caption:= 'Template Form Tester';
23182:   FormStyle:= fsStayOnTop;
23183:   with editor do begin
23184:     Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mx2.rtf'
23185:     SelStart:= 0;
23186:     Modified:= False;
23187:   end;
23188: end;
23189: with TWebMainForm.create(self) do begin
23190:   URLs.Text:= 'http://www.kleiner.ch';
23191:   URLsClick(self); Show;
23192: end;
23193: with TSynexportForm.create(self) do begin
23194:   Caption:= 'Synexport HTML RTF tester';
23195:   Show;
23196: end;
23197: with TThreadSortForm.create(self) do begin
23198:   ShowModal; free;
23199: end;
23200: with TCustomDrawForm.create(self) do begin
23201:   width:=820; height:=820;
23202:   imagel.height:= 600; //add properties
23203:   imagel.picture.bitmap:= image2.picture.bitmap;
23204:   //SelectionBackground1Click(self) CustomDraw1Click(self);
23205:   Background1.click;
23206:   bitmap1.click; Tile1.click;
23207:   ShowModal;
23208:   Free;
23209: end;
23210: with TfplotForm1.Create(self) do begin
23211:   BtnPlotClick(self);
23212:   ShowModal; Free;
23213: end;
23214: with TOvcCalculator.create(self) do begin
23215:   parent:= aForm; //free;
23216:   setbounds(550,510,200,150);
23217:   displaystr:= 'maXcalc';
23218: end;
23219: with THexForm2.Create(self) do begin
23220:   ShowModal;
23221:   Free;
23222: end;
23223:
23224: function CheckBox: string;
23225: var idHTTP: TIDHTTP;
23226: begin
23227:   result:= 'version not found';
23228:   if IsInternet then begin
23229:     idHTTP:= TIdHTTP.Create(NIL);
23230:     try
23231:       result:= idHTTP.Get(MXVERSIONFILE2);
23232:       result:= result[1]+result[2]+result[3]+result[4]+result[5];
23233:       if result = MBVER2 then begin
23234:         //Speak(' A new Version '+vstr+' of max box is available! ');
23235:         result:= ('!!!! OK. You have latest Version: '+result+' available at '+MXSITE);
23236:       end; //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
23237:     finally
23238:       idHTTP.Free
23239:     end;
23240:   end;
23241: end;
23242:
23243: //Runtime Spec Functions Edition 190
23244: function ApWinExecAndWait32(FileName:PChar; CommandLine:PChar; Visibility:Integer):Integer;
23245: function KillTask(ExeFileName: string): Integer;
23246: procedure KillProcess(hWindowHandle: HWND);
23247: function FindWindowByTitle(WindowTitle: string): Hwnd;
23248: function IntToFloat(i: Integer): double;
23249: function AddThousandSeparator(S: string; myChr: Char): string;
23250: function mciSendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
23251: procedure FormAnimation(Sender: TObject; adelay: integer);
23252: procedure LoadResourceFile2(aFile:string; ms:TMemoryStream);
23253: function putBinResTo(binresname: pchar; newpath: string): boolean;
23254: procedure ExecuteHyperlink(Sender: TObject; HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:string);
23255: function IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:string; MouseX,MouseY:Integer;var HyperLink:string):Bool
23256: Function GetWindowThreadProcessId( hWnd : HWND; var dwProcessId : DWORD ) : DWORD; '
23257: Function GetWindowTask( hWnd : HWND ) : THandle';
23258: Function LoadBitmap( hInstance : HINST; lpBitmapName : PChar ) : HBITMAP';
23259: Function GetCommConfig(hCommDev: THandle; var lpCC: TCommConfig; var lpdwSize: DWORD): BOOL';
23260: function WinExecAndWait32(FileName: string; Visibility: Integer): Longword;
23261: Function MakeHash( const s : TbtString ) : Longint';
23262: Function GetUsedUnitList( list : Tstringlist ) : string';
23263: function ConsoleCapture(const _dirName, _exeName, _cmdline: string; amemo: TStringlist): Boolean;';
23264: function ConsoleCaptureDOS(const _dirName, _exeName, _cmdLine: string; amemo: TStrings): Boolean;';
23265:   srlist:= TStringlist.create;

```

```

23266:      ConsoleCapture('C:\', 'cmd.exe', '/c dir *.*', srlst);
23267:      writeln(srlst.text); srlst.Free;
23268:  function RunCaptured(const _dirName, _exeName, _cmdLine: string; amemo: TStringlist): Boolean;';
23269:  Function SamePropTypeNames( const Name1, Name2 : ShortString ) : Boolean';
23270:  Function FloatToStrEx( Value : Extended ) : string';
23271:  Function StrToFloatEx( const S : string ) : Extended';
23272:  Procedure PerformanceDelayMS(ams: integer); //microsecond resolution delay!
23273: //http://www.swissdelphicenter.ch/en/showcode.php?id=2179
23274:  function ExecuteProcess(FileName: string;Visibility:Integer; BitMask:Integer; Synch:Boolean):Longword;
23275:  Function
23276:    ExecuteMultiProcessor(FileName:string;Visibility:Integer;BitMask:Integer;Synch:Boolean):Longword;';
23276:    if ExecuteMultiProcessor('notepad.exe', SW_SHOW, 2, true) = 0 then
23277:      writeln('Multiprocessing Runs on CPU 2');
23278:  procedure StartServiceAfterInstall(aservy: TService);
23279:  Function GetDllVersion2(DllName: string; var DLLVersionInfo: TDLLVersionInfo): Boolean;
23280:  procedure SendCopyMessage(amess, astation: string); //communicate process-spanned with WM_COPYDATA
23281:  function BrowseComputer2(DialogTitle: string; var CompName: string; bNewStyle: Boolean): Boolean;
23282:  function ChangeAlphaTo(input: string; aoffset: byte): string';
23283:  function CheckIBAN(iban: string): Boolean';
23284:  function CreateIDStack; //instance to IdStack of CreateStack
23285:  Function GetRecordCount(DataSet: TBDEDataSet): Longint;';
23286:  Function CountPos(const subtxt: string; Text: string): Integer;
23287:  procedure HTMLOtoRTF(html: string; var rtf: TRichedit);
23288:  procedure ReversePlay(const szFileName: string);
23289:  Function ADOConnectionString(ParentHandle:THandle; InitialString:WideString;out NewString: string):Boolean;
23290:  procedure ShowEOleException(AExc: EOleException; Query: String);
23291:  function UpdateBlob(Connection: TADOConnection; Spalte: String; Tabelle: String; Where: String; var ms: TMemoryStream): Boolean;
23292: .. save HTML pages as MHTML (HTML Archiv Format)
23293: http://www.swissdelphicenter.ch/en/showcode.php?id=2300
23294:  function SaveToMHT(const AUrl, AFileName: string; AShowErrorMessage: Boolean = False): Boolean;
23295:  function FileType2MimeType(const AFileName: string): string;
23296:  function DownloadURL_NOCache(const aUrl: string; var s: String): Boolean;
23297:  Function IsCOMObjectActive(Classname: string): Boolean;
23298:  Procedure CopyHTMLToClipboard(const str: string; const htmlStr: string = '');
23299:  function CheckCreditCard(c: string): Integer;
23300:  0: Card is invalid or unknown 1: is a AmEx 2: is a Visa 3: is a valid MasterCard
23301:
23302:  function getFormRes(classname: string): string; //shows DFM Res of Exe!
23303:  procedure OutputDebugString(PChar(Format('[%s][%s] %s',[aCaption, GetFormatDT(StartDT), aText])));
23304:  procedure ScanNetworkResources(ResourceType, DisplayType: DWord; List: TStrings);
23305: //ScanNetworkResources(RESOURCETYPE_DISK, RESOURCEDISPLAYTYPE_SERVER, ListBox1.Items);
23306:
23307:  procedure SQLDropField(dbName, tblName, fldName: String); {Field Name to Drop}
23308:  type TCastType = (ctSmallInt, ctInteger, ctDecimal, ctNumeric, ctFloat,
23309:                      ctChar, ctVarChar, ctDate, ctBoolean, ctBLOB, ctTime,
23310:                      ctTimeStamp, ctMoney, ctAutoInc, ctBytes);
23311: {Blob definition type 1 = Memo, 2 = Binary, 3 = Formatted Memo, 4 = OLE Object, 5 = Graphic}
23312:
23313:  procedure SQLAddField(dbName,tblName,fldName: String; fldType: TCastType; fldLength, precisOrBlobLen,
23314: scaleOrBlobType: Integer);
23314: const
23315:  UrlGeoLookupInfo ='http://ipinfodb.com/ip_query.php?timezone=true&ip=%s';
23316:  UrlGeoLookupInfo2 ='http://backup.ipinfodb.com/ip_query.php?timezone=true&ip=%s'; //backup
23317:
23318:  procedure GetGeoInfo(const IPAddress : string;var GeoInfo : TGeoInfo; const UrlGeoLookupInfo: string);
23319:  CL.AddTypeS('TGeoInfo', 'record status: string; countrycode : '
23320:    +string; countryname : string; regioncode : string; city : string; zippostalcode : string; latitude '
23321:    +: string; longitude : string; timezonename : string; gmtoffset: string; isdst: string; end');
23322:
23323:  procedure SIRegister_ubigFloatV3(CL: TPSPascalCompiler);
23324: begin
23325:   CL.AddTypeS('TMaxSig', 'integer');
23326:   CL.AddTypeS('TView', '( normal, Scientific )');
23327:   SIRegister_TFloatInt(CL); SIRegister_TBigFloat(CL);
23328: end;
23329:
23330:  procedure SIRegister_UBigIntsV4(CL: TPSPascalCompiler);
23331: begin TDigits', 'array of int64'); SIRegister_TInteger(CL);
23332:  Procedure SetBaseVal( const newbase : integer)');
23333:  Function GetBasePower : integer');
23334:  Function GetBase : integer');
23335:  Procedure SetThreadSafe( newval : boolean)');
23336: end;
23337:
23338:  function BigDiv(aone, atwo: string): string;
23339:  var tbig1, tbig2: TInteger;
23340:  begin
23341:    tbig1:= TInteger.create(10);
23342:    tbig2:= TInteger.create(10);
23343:    try
23344:      tbig1.assign2(atwo)
23345:      tbig2.assign2(aone)
23346:      tbig2.Divide(tbig1)
23347:    finally
23348:      result:= tbig2.Convert.ToDecimalString(false)
23349:      tbig1.Free;
23350:      tbig2.free;
23351:    end;

```

```

23352: end;
23353:
23354: procedure SIRegister_UDict(CL: TPSPascalCompiler);
23355: begin CL.AddConstantN('dichighletter','String').SetString( 'z' );
23356:   SIRegister_TDicForm(CL);
23357:   SIRegister_TDic(CL);
23358: end;
23359:
23360: procedure SetArrayLength2Char2(var arr: T2CharArray; asizel, asize2: integer);
23361: var i: integer;
23362: begin setlength(arr, asizel);
23363:   for i:= 0 to asizel do SetLength(arr[i], asize2);
23364: end;
23365:
23366: procedure SIRegister_UP10Build(CL: TPSPascalCompiler);
23367: begin
23368:   Procedure ParseFunction(FunctionString: string; Variables: TStringlist; FunctionOne, FunctionTwo:
TStringList; UsePascalNumbers : boolean; var FirstTOP : {TObject}POperation; var Error : boolean)');
23369: end;
23370:
23371: TDLLVersionInfo=Record
23372:   cbSize, // Size of the structure, in bytes.
23373:   dwMajorVersion, // Major version of the DLL
23374:   dwMinorVersion, // Minor version of the DLL
23375:   dwBuildNumber, // Build number of the DLL
23376:   dwPlatformID: DWord; // Identifies the platform for which the DLL was built
23377: end;
23378:
23379: command1:= 'play "'+songpath+'maxbox.wav"';
23380: command2:= 'play "'+songpath+'moon.wav"';
23381: SendMCICmd('open waveaudio shareable'); //parallels
23382: SendMCICmd('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\maxbox.wav"');
23383: SendMCICmd('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\moon.wav"');
23384: SendMCICmd('close waveaudio');
23385:
23386:
23387: /////////////////////////////////
23388: All maXbox Tutorials Table of Content 2014/2015
23389: /////////////////////////////////
23390: Tutorial 00 Function-Coding (Blix the Programmer)
23391: Tutorial 01 Procedural-Coding
23392: Tutorial 02 OO-Programming
23393: Tutorial 03 Modular Coding
23394: Tutorial 04 UML Use Case Coding
23395: Tutorial 05 Internet Coding
23396: Tutorial 06 Network Coding
23397: Tutorial 07 Game Graphics Coding
23398: Tutorial 08 Operating System Coding
23399: Tutorial 09 Database Coding
23400: Tutorial 10 Statistic Coding
23401: Tutorial 11 Forms Coding
23402: Tutorial 12 SQL DB Coding
23403: Tutorial 13 Crypto Coding
23404: Tutorial 14 Parallel Coding
23405: Tutorial 15 Serial RS232 Coding
23406: Tutorial 16 Event Driven Coding
23407: Tutorial 17 Web Server Coding
23408: Tutorial 18 Arduino System Coding
23409: Tutorial 18_3 RGB LED System Coding
23410: Tutorial 19 WinCOM /Arduino Coding
23411: Tutorial 20 Regular Expressions RegEx
23412: Tutorial 21 Android Coding (coming 2013)
23413: Tutorial 22 Services Programming
23414: Tutorial 23 Real Time Systems
23415: Tutorial 24 Clean Code
23416: Tutorial 25 maXbox Configuration I+II
23417: Tutorial 26 Socket Programming with TCP
23418: Tutorial 27 XML & TreeView
23419: Tutorial 28 DLL Coding (available)
23420: Tutorial 29 UML Scripting (2014)
23421: Tutorial 30 Web of Things (2014)
23422: Tutorial 31 Closures (2014)
23423: Tutorial 32 SQL Firebird (2014)
23424: Tutorial 33 Oscilloscope (available)
23425: Tutorial 34 GPS Navigation (2014)
23426: Tutorial 35 Web Box (available)
23427: Tutorial 36 Unit Testing (coming 2015)
23428: Tutorial 37 API Coding (coming 2015)
23429: Tutorial 38 3D Coding (coming 2015)
23430: Tutorial 39 GEO Map Coding (available)
23431: Tutorial 39_1 GEO Map Layers Coding (available)
23432: Tutorial 40 REST Coding (coming 2015)
23433: Tutorial 41 Big Numbers Coding (coming 2015)
23434: Tutorial 42 Parallel Processing (coming 2015)
23435:
23436:
23437: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
23438: using Docu for this file is maxbox_functions_all.pdf
23439: PEP - Pascal Education Program Low Lib Lab ShellHell in UDEMY

```

```

23440:
23441: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials
23442: http://stackoverflow.com/tags/pascalscript/hot
23443: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
23444: http://sourceforge.net/projects/maxbox #locs:51620
23445: http://sourceforge.net/apps/mediawiki/maxbox
23446: http://www.blaisepascal.eu/
23447: https://github.com/maxkleiner/maxbox3.git
23448: http://www.heise.de/download/maxbox-1176464.html
23449: http://www.softpedia.com/get/Programming/Other-Programming-Files/maxbox.shtml
23450: https://www.facebook.com/pages/Programming-maXbox/166844836691703
23451: http://www.softwareschule.ch/arduino_training.pdf
23452: http://www.delphairea.com
23453: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
23454: http://entwickler-konferenz.de/2014/speakers/max-kleiner
23455: http://www.heise.de/download/maxbox-1176464.html
23456: https://www.udemy.com/learn-coding-from-the-scratch
23457: http://www.slideshare.net/maxkleiner1/codesign-2015
23458: https://www.dropbox.com/s/yolconwmq4ogta4/Blaise_2and3_SP_Total.pdf?dl=0
23459: http://www.softwareschule.ch/download/maxbox_promo.png
23460: http://max.kleiner.com/maxbox_functions_all.htm
23461: http://max.kleiner.com/boxart.htm
23462: http://www.jurgott.org/linkage/util.htm
23463: http://www.swissdelphicenter.ch/en/niklauswirth.php
23464: http://www.softwareschule.ch/images/maxbox_20years_delphi.jpg
23465:
23466:
23467: All maxbox Examples List
23468: https://github.com/maxkleiner/maxbox3/releases
23469: ****
23470: 000_pas_baseconvert.txt
23471: 000_pas_baseconvert.txt_encrypt
23472: 000_pas_baseconvert.txt_decrypt
23473: 001_1_pas_functest - Kopie.txt
23474: 001_1_pas_functest.txt
23475: 001_1_pas_functest2.txt
23476: 001_1_pas_functest_clx2.txt
23477: 001_1_pas_functest_clx2_2.txt
23478: 001_1_pas_functest_openarray.txt
23479: 001_pas_lottogen.txt
23480: 001_pas_lottogen_template.txt
23481: 001_pas_lottogen_txtcopy
23482: 002_pas_russianroulette.txt
23483: 002_pas_russianroulette.txtcopy
23484: 002_pas_russianroulette.txtcopy_decrypt
23485: 002_pas_russianroulette.txtcopy_encrypt
23486: 003_pas_motion.txt
23487: 003_pas_motion.txtcopy
23488: 004_pas_search.txt
23489: 004_pas_search_replace.txt
23490: 004_search_replace_allfunctionlist.txt
23491: 005_pas_oodesign.txt
23492: 005_pas_shelllink.txt
23493: 006_pas_oobatch.txt
23494: 007_pas_streamcopy.txt
23495: 008_EINMALEINS_FUNC.TXT
23496: 008_explanation.txt
23497: 008_pas_verwechselt.txt
23498: 008_pas_verwechselt_ibz_bern_func.txt
23499: 008_stack_ibz.TXT
23500: 009_pas_umrunner.txt
23501: 009_pas_umrunner_all.txt
23502: 009_pas_umrunner_componenttest.txt
23503: 009_pas_umrunner_solution.txt
23504: 009_pas_umrunner_solution_2step.txt
23505: 010_pas_oodesign_solution.txt
23506: 011_pas_puzzelpas_defect.txt
23507: 012_pas_umrunner_solution.txt
23508: 012_pas_umrunner_solution2.txt
23509: 013_pas_linenumber.txt
23510: 014_pas_primetest.txt
23511: 014_pas_primetest_first.txt
23512: 014_pas_primetest_sync.txt
23513: 015_pas_designbycontract.txt
23514: 015_pas_designbycontract_solution.txt
23515: 016_pas_searchrec.txt
23516: 017_chartgen.txt
23517: 018_data_simulator.txt
23518: 019_dez_to_bin.txt
23519: 019_dez_to_bin_grenzwert_ibz.txt
23520: 020_proc_feedback.txt
23521: 021_pas_symkey.txt
23522: 021_pas_symkey_solution.txt
23523: 022_pas_filestreams.txt
23524: 023_pas_find_searchrec.txt
23525: 023_pas_pathfind.txt
23526: 024_pas_TFileStream_records.txt
23527: 025_prime_direct.txt
23528: 026_pas_memorystream.txt
282_fadengraphik.txt
283_SQL_API_messagetimeout.txt
284_SysTools4.txt
285_MineForm_GR32.TXT
285_MineForm_GR32main.TXT
285_MineForm_GR32mainsolution.TXT
285_MineForm_propas.TXT
285_MineForm_propas2.TXT
285_minesweeper2.TXT
285_Patterns_process.txt
286_colormixer_jpeg_charcounter.txt
286_colormixer_jpeg_charcounter2.txt
287_eventhandling.txt
287_eventhandling2.txt
287_eventhandling2_negpower.txt
288_bitblt.txt
288_bitblt_resize.txt
289_regression.txt
289_regression2.txt
290_bestofbox.txt
290_bestofbox2.txt
290_bestofbox3.txt
291_3sort_visual_thread.txt
292_refactoring2.txt
293_bold_utils.txt
293_ib_utils.txt
293_ib_utils_timetest.txt
294_maxcalc_demo.txt
294_maxcalc_demo2.txt
295_easter_calendar.txt
295_easter_calendar2.txt
295_easter_combobox.txt
297_atomimage.txt
297_atomimage2.txt
297_atomimage3.txt
297_atomimage4.txt
297_maxonmotor.TXT
297_maxon_atomimage9.txt
298_bitblt_animation.txt
298_bitblt_animation2.txt
298_bitblt_animation3.txt
298_bitblt_animation4.txt
298_bitblt_animation4_screensaver.txt
298_bitblt_animation5_screensaver.txt
299_animation.txt
299_animationmotor_arduino.txt
299_animation_formprototype.txt
299_realtimeclock_arduino.txt
299_realtimeclock_arduino2.txt
300_treeview.txt
300_treeview_test.txt
300_treeview_test2.txt
300_treeview_test3.txt
301_LED_Arduino3.txt
301_led_arduino3_simple.txt
301_led_arduino3_simplecode.txt
301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt

```

```

23529: 027_pas_shelleexecute_beta.txt
23530: 027_pas_shelleexecute_solution.txt
23531: 028_pas_dataset.txt
23532: 029_pas_assignfile.txt
23533: 029_pas_assignfile_dragndropexe.txt
23534: 030_palindrome_2.txt
23535: 030_palindrome_tester.txt
23536: 030_pas_recursion.txt
23537: 030_pas_recursion2.txt
23538: 031_pas_hashcode.txt
23539: 032_pas_crc_const.txt
23540: 033_pas_cipher.txt
23541: 033_pas_cipher_def.txt
23542: 033_pas_cipher_file_2_solution.txt
23543: 034_pas_soundbox.txt
23544: 035_pas_crcscript.txt
23545: 035_pas_CRCscript_modbus.txt
23546: 036_pas_includetest.txt
23547: 036_pas_includetest_basta.txt
23548: 037_pas_define_demo32.txt
23549: 038_pas_box_demonstrator.txt
23550: 039_pas_dllcall.txt
23551: 040_paspointer.txt
23552: 040_paspointer_old.txt
23553: 041_pasplotter.txt
23554: 041_pasplotter_plus.txt
23555: 042_pas_kgv_ggt.txt
23556: 043_pas_proceduretype.txt
23557: 044_pas_l4queens_solvith14.txt
23558: 044_pas_8queens.txt
23559: 044_pas_8queens_sol2.txt
23560: 044_pas_8queens_solutions.txt
23561: 044_queens_performer.txt
23562: 044_queens_performer2.txt
23563: 044_queens_performer2tester.txt
23564: 045_pas_listhandling.txt
23565: 046_pas_records.txt
23566: 047_pas_modula10.txt
23567: 048_pas_romans.txt
23568: 049_pas_ifdemo.txt
23569: 049_pas_ifdemo_BROKER.txt
23570: 050_pas_primetest2.txt
23571: 050_pas_primestester_thieves.txt
23572: 050_program_starter.txt
23573: 050_program_starter_performance.txt
23574: 051_pas_findtext_solution.txt
23575: 052_pas_text_as_stream.txt
23576: 052_pas_text_as_stream_include.txt
23577: 053_pas_singleton.txt
23578: 054_pas_speakpassword.txt
23579: 054_pas_speakpassword2.txt
23580: 054_pas_speakpassword_searchtest.txt
23581: 055_pas_factorylist.txt
23582: 056_pas_demeter.txt
23583: 057_pas_dirfinder.txt
23584: 058_pas_filefinder.txt
23585: 058_pas_filefinder_pdf.txt
23586: 058_pas_filefinder_screview.txt
23587: 058_pas_filefinder_screview2.txt
23588: 058_pas_filefinder_screview3.txt
23589: 059_pas_timertest.txt
23590: 059_pas_timertest_2.txt
23591: 059_pas_timertest_time_solution.txt
23592: 059_timerobject_starter2.txt
23593: 059_timerobject_starter2_ibz2_async.txt
23594: 059_timerobject_starter2_uml.txt
23595: 059_timerobject_starter2_uml_main.txt
23596: 059_timerobject_starter4_ibz.txt
23597: 060_pas_datefind.txt
23598: 060_pas_datefind_exceptions2.txt
23599: 060_pas_datefind_exceptions_CHECKTEST.txt
23600: 060_pas_datefind_fulltext.txt
23601: 060_pas_datefind_plus.txt
23602: 060_pas_datefind_plus_mydate.txt
23603: 061_pas_randomwalk.txt
23604: 061_pas_randomwalk_plus.txt
23605: 062_paskorrelation.txt
23606: 063_pas_calculateform.txt
23607: 063_pas_calculateform_2list.txt
23608: 064_pas_timetest.txt
23609: 065_pas_bitcounter.txt
23610: 066_pas_eliza.txt
23611: 066_pas_eliza_include_sol.txt
23612: 067_pas_morse.txt
23613: 068_pas_piezo_sound.txt
23614: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
23615: 069_my_LEDBOX.TXT
23616: 069_pas_ledmatrix.txt
23617: 069_pas_LEDMATRIX_Alphabet.txt

301_SQL_DBfirebird4.txt
302_LCLActivity_java.txt
302_LED_DataLogger.txt
303_Android_LCLActivity_java.txt
303_webserver.txt
303_webserver2.txt
303_webserver_alldocs2.txt
303_webserver_alldocs2_tester.txt
303_webserver_minimal.txt
303_webserver_simple.txt
304_st_system.txt
305_indy_elizahttpserver.TXT
305_indy_elizahttpserver2.TXT
305_indy_elizahttpserver3.TXT
305_indy_elizahttpserver4file.TXT
305_webserver_arduino.txt
305_webserver_arduino2.txt
305_webserver_arduino3.txt
305_webserver_arduino3ibz.txt
305_webserver_arduino3ibz_rgb_led.txt
305_webserver_arduino3test.txt
306_SPS_http_command.txt
307_all_booleanlogic.txt
308_bitbox3.txt
308_bitbox3_exec.txt
308_boolean_animation.txt
308_boolean_animation2.txt
309_regex_power.txt
309_regex_powertester2.txt
309_regex_powertester3.txt
310_regex_decorator.TXT
312_ListView.txt
313_dmath_dll.txt
314_fundamentals4_tester.TXT
315_funcplot_dmath.TXT
316_cfileutils_cdatetime_tester.TXT
317_excel_export_tester.TXT
318_excel_export.TXT
318_excel_export2.TXT
318_excel_export3.TXT
318_excel_export4.TXT
318_excel_export3_tester.TXT
319_superfunctions_math.TXT
319_superfunctions_mathdefect.TXT
320_superfunctions.TXT
320_superfunctions2.TXT
321_SQL_Excel.txt
321_SQL_Excel2.txt
321_SQL_Excel_Export.txt
321_SQL_ExportExec.txt
321_SQL_ExportTest.txt
321_SQL_SAS_tester3.txt
321_SQL_SAS_tester3_selfcompile.txt
321_SQL_SAS_tester3_selfcompile2.txt
321_SQL_SAS_tester4.txt
321_SQL_SAS_updater.txt
322_timezones.TXT
323_datefind_fulltext_search.txt
323_datefind_fulltext_searchtester.txt
324_interfacenavi.TXT
325_ampelsteuerung.txt
325_analogclock.txt
326_world_analogclock.txt
326_world_analogclock2.txt
327_atomimage_clock.txt
328_starfield.txt
329_starfield2.txt
330_myclock.txt
330_myclock2.txt
331_SQL_DBfirebird4.txt
332_jprofiler.txt
332_jprofiler_form.txt
332_jprofiler_form2.txt
333_querybyexample.txt
333_querybyexample2.txt
334_jvutils_u.txt
335_atomimage5.txt
335_atomimage6.txt
335_atomimage7.txt
336_digiclock.txt
336_digiclock2.txt
336_digiclock2test.txt
336_digiclock3.txt
337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt
339_ntfs.txt
340_docutype.txt
340_logsimulation.txt

```

```

23618: 069_pas_LEDMATRIX_Alphabet_run.txt          340_logsimulation2.txt
23619: 069_pas_LEDMATRIX_Alphabet_tester.txt       340_soundControltype.txt
23620: 069_PAS_LEDMATRIX_COLOR.TXT                 341_blix_clock.txt
23621: 069_pas_ledmatrix_fixedit.txt              341_blix_clock2.txt
23622: 069_pas_LEDMATRIX_soundbox.txt             341_blix_clock_tester.txt
23623: 069_pas_LEDMATRIX_soundbox2.txt            342_set_enumerator.txt
23624: 069_Richter_MATRIX.TXT                     343_dice2.txt
23625: 070_pas_functionplot.txt                   344_pe_header.txt
23626: 070_pas_functionplotter2.txt               344_pe_header2.txt
23627: 070_pas_functionplotter2_mx4.txt           345_velocity.txt
23628: 070_pas_functionplotter2_tester.txt        346_conversions.txt
23629: 070_pas_functionplotter3.txt               347_pictureview.txt
23630: 070_pas_functionplotter4.txt               348_duallistview.txt
23631: 070_pas_functionplotter_digital.txt        349_biginteger.txt
23632: 070_pas_functionplotter_elliptic.txt       350_parserform.txt
23633: 070_pas_function_helmholtz.txt            351_chartform.txt
23634: 070_pas_textcheck_experimental.txt         351_chartform2.txt
23635: 071_pas_graphics.txt                      351_chartform3.txt
23636: 071_pas_graphics_drawsym.txt              352_array_unittest.txt
23637: 071_pas_graphics_drawsym_save.txt          353_smtp_email.txt
23638: 071_pas_graphics_random.txt               353_smtp_email2.txt
23639: 072_pas_fractals.txt                      354_josephus.txt
23640: 072_pas_fractals_2.txt                    355_life_of_PI.txt
23641: 072_pas_fractals_blackhole.txt             356_3D_printer.txt
23642: 072_pas_fractals_perfomance.txt           357_fplot.TXT
23643: 072_pas_fractals_performance_new.txt       358_makesound.txt
23644: 072_pas_fractals_perfomance_sharp.txt      359_charsetrules.TXT
23645: 072_pas_fractals_performance.txt           360_allobjects.TXT
23646: 072_pas_fractals_performance_mx4.txt        360_JvPaintFX.TXT
23647: 073_pas_forms.txt                         361_heartbeat_wave.TXT
23648: 074_pas_chartgenerator.txt                 362_maxonmotor2.TXT
23649: 074_pas_chartgenerator_solution.txt        363_compress_services.txt
23650: 074_pas_chartgenerator_solution_back.txt   363_compress_services2.txt
23651: 074_pas_charts.txt                        364_pdf_services.txt
23652: 075_bitmap_Artwork2.txt                   365_memorystream.txt
23653: 075_pas_bitmappuzzle.txt                  365_memorystream2.txt
23654: 075_pas_bitmappuzzle24.prod.txt           365_memorystream_test.txt
23655: 075_pas_bitmappuzzle2_prod.txt             365_U_HexView.txt
23656: 075_pas_bitmappuzzle3.txt                 366_mp3player.txt
23657: 075_pas_bitmapsolve.txt                   366_mp3player2.txt
23658: 075_pas_bitmap_Artwork.txt                366_mp3player2_themestest.txt
23659: 075_pas_puzzlepas_solution.txt            367_silvi_player_widgets.txt
23660: 076_pas_3dcube.txt                        367_silvi_player_widgets2.txt
23661: 076_pas_circle.txt                       367_widgets.txt
23662: 077_pas_mmshow.txt                        368_configuration_demo.txt
23663: 078_pas_pi.txt                           369_macro_demo.txt
23664: 079_pas_3dcube_animation.txt              370_callback2grid.TXT
23665: 079_pas_3dcube_animation4.txt             370_richedit.txt
23666: 079_pas_3dcube_plus.txt                  370_richedit_highlight.txt
23667: 080_pas_hanoi.txt                        370_synedit.txt
23668: 080_pas_hanoi2.txt                       370_synedit2.txt
23669: 080_pas_hanoi2_file.txt                  370_synedit2_mxtester.txt
23670: 080_pas_hanoi2_sol.txt                   370_synedit2_mxtester2.txt
23671: 080_pas_hanoi2_tester.txt                371_maxbook_v4tester.txt
23672: 080_pas_hanoi2_tester_fast.txt            372stackibz2_memoryalloc.TXT
23673: 080_pas_hanoi3.txt                       372_synedit_export.txt
23674: 081_pas_chartist2.txt                    373_batman.txt
23675: 082_pas.biorythmus.txt                   373_fractals_tvout.txt
23676: 082_pas.biorythmus_solution.txt           374_realtime_random.txt
23677: 082_pas.biorythmus_solution_3.txt         374_realtime_random2.txt
23678: 082_pas.biorythmus_test.txt              374_realtime_randomtest.txt
23679: 083_pas_GITARRE.txt                      374_realtime_randomtest2.txt
23680: 083_pas_soundbox_tones.txt                375_G9_musicbox.txt
23681: 084_pas_waves.txt                        376_collections_list.txt
23682: 085_mxsinus_logo.txt                     377_simpleXML.txt
23683: 085_sinus_plot_waves.txt                 377_smartXML.txt
23684: 086_pas_graph_arrow_heart.txt             377_smartXMLWorkshop.txt
23685: 087_bitmap_loader.txt                    377_smartXMLWorkshop2.txt
23686: 087_pas_bitmap_solution.txt              378_queryperformance3.txt
23687: 087_pas_bitmap_solution2.txt              378_REST1.txt
23688: 087_pas_bitmap_subimage.txt              378_REST2.txt
23689: 087_pas_bitmap_test.txt                  379_timefunc.txt
23690: 088_pas_soundbox2_mp3.txt                379_timefuncTesterfilemon.txt
23691: 088_pas_soundbox_mp3.txt                 380_coolfunc.txt
23692: 088_pas_sphere_2.txt                     380_coolfunc2.txt
23693: 089_pas_gradient.txt                    380_coolfunc_tester.txt
23694: 089_pas_maxland2.txt                     381_bitcoin_simulation.txt
23695: 090_pas_sudoku4.txt                      382_GRMath.TXT
23696: 090_pas_sudoku4_2.txt                   382_GRMath_PI_Proof.TXT
23697: 091_pas_cube4.txt                        382_GRMath_Riemann.TXT
23698: 092_pas_statistics4.txt                 383_MDAC_DCOM.txt
23699: 093_variance.txt                         384_TeamViewerID.TXT
23700: 093_variance_debug.txt                   386_InternetRadio.TXT
23701: 094_pas_daysold.txt                     387_fulltextfinder.txt
23702: 094_pas_stat_date.txt                   387_fulltextfinder_cleancode.txt
23703: 095_pas_ki_simulation.txt               387_fulltextfinder_fast.txt
23704: 096_pas_geisen_problem.txt              387_fulltext_getscripttest.txt
23705: 096_pas_montyhall_problem.txt            388_TCPServerSock.TXT
23706: 097_lotto_proofofconcept.txt             388_TCPServerSock2.TXT

```

```

23707: 097_pas_lottocombinations_beat_plus.txt          388_TCPServerSockClient.TXT
23708: 097_pas_lottocombinations_beat_plus2.txt        389_TAR_Archive.TXT
23709: 097_pas_lottocombinations_universal.txt         389_TAR_Archive_test.TXT
23710: 097_pas_lottosimulation.txt                     389_TAR_Archive_test2.TXT
23711: 098_pas_chartgenerator_plus.txt                 390_Callback3.TXT
23712: 099_pas_3D_show.txt                           390_Callback3Rec.TXT
23713: 200_big_numbers.txt                          390_CallbackClean.TXT
23714: 200_big_numbers2.txt                         390_StringlistHTML.TXT
23715: 201_streamload_xml.txt                      391_ToDo_List.TXT
23716: 202_systemcheck.txt                        392_Barcodex.TXT
23717: 203_webservice_simple_intftester.txt       392_Barcodex2.TXT
23718: 204_webservice_simple.txt                  392_Barcodex23.TXT
23719: 205_future_value_service.txt               392_Barcodex2scholz.TXT
23720: 206_DTD_string_functions.txt             392_Barcodex3scholz.TXT
23721: 207_ibz2_async_process.txt                393_QRCode.TXT
23722: 208_crc32_hash.txt                       393_QRCode2.TXT
23723: 209_cryptohash.txt                        393_QRCode2Direct.TXT
23724: 210_public_private.txt                   393_QRCode2DirectIndy.TXT
23725: 210_public_private_cryptosystem.txt     393_QRCode2Direct_detlef.TXT
23726: 211_wipe_pattern.txt                     393_QRCode3.TXT
23727: 211_wipe_pattern2.txt                   394_networkgraph.TXT
23728: 211_wipe_pattern_solution.txt          394_networkgraph_depwalkutilstest.TXT
23729: 212_pas_statisticmodule4.TXT           394_networkgraph_depwalkutilstest2.TXT
23730: 212_pas_statisticmoduletxt.TXT         395_USBController.TXT
23731: 212_statisticmodule4.txt                396_Sort.TXT
23732: 213_pas_BBP_Algo.txt                   397_Hotlog.TXT
23733: 214_mxdocudemo.txt                     397_Hotlog2.TXT
23734: 214_mxdocudemo2.txt                   398_ustrings.txt
23735: 214_mxdocudemo3.txt                   399_form_templates.txt
23736: 215_hints_test.TXT                     400_fploottchart.TXT
23737: 216_warnings_test.TXT                 400_fploottchart2.TXT
23738: 217_pas_heartbeat.txt                 400_fploottchart2teetest.TXT
23739: 218_biorhythmus_studio.txt          400_QRCodeMarket.TXT
23740: 219_cipherbox.txt                     401_tfilerun.txt
23741: 219_crypt_source_comtest_solution.TXT 402_richedit2.txt
23742: 220_cipherbox_form.txt              403_outlookspy.txt
23743: 220_cipherbox_form2.txt            404_simplebrowser.txt
23744: 221_bcd_explain.txt                405_datefinder_today.txt
23745: 222_memoform.txt                  406_portscan.txt
23746: 223_directorybox.txt              407_indydemo.txt
23747: 224_dialogs.txt                   408_testroboter.txt
23748: 225_sprite_animation.txt          409_excel_control.txt
23749: 226_ASCII_Grid2.TXT              410_keyboardevent.txt
23750: 227_animation.txt                411_json_test.txt
23751: 227_animation2.txt              412_Zeosutils.txt
23752: 228_android_calendar.txt        413_listview2.txt
23753: 229_android_game.txt            414_avrdude_flash.txt
23754: 229_android_game_tester.txt    415_avrdude_writehex.txt
23755: 230_DataProvider.txt           416_sonar_startscriptEKON.TXT
23756: 230_DataSetProvider.txt        416_sonar_startscriptEKON_reporting.TXT
23757: 230_DataSetXMLBackupScholz.txt 417_GRMath_PI_Proof2.TXT
23758: 231_DBGrid_access.txt          418_functional_paradigm.txt
23759: 231_DBGrid_XMLaccess.txt      419_archimedes_spiral.txt
23760: 231_DBGrid_XMLaccess2.txt    419_archimedes_spiral2.txt
23761: 231_DBGrid_XMLaccess_locatesteller.txt 420_archimedes_arduino.txt
23762: 231_DBGrid_XML_CDS_local.txt 420_Lissajous.txt
23763: 232_outline.txt                421_PI_Power.TXT
23764: 232_outline_2.txt              421_PI_Power2.TXT
23765: 233_modular_form.txt          422_world_bitboxx.txt
23766: 234_debugoutform.txt         423_game_of_life.TXT
23767: 235_fastform.TXT             423_game_of_life2.TXT
23768: 236_componentpower.txt       423_game_of_life3.TXT
23769: 236_componentpower_back.txt  423_game_of_life3_test.TXT
23770: 237_pas_4forms.txt           423_game_of_life4.TXT
23771: 238_lottogen_form.txt        423_game_of_life4_kryptum.TXT
23772: 239_pas_sierpinskii.txt      424_opengl_tester.txt
23773: 239_pas_sierpinskii2.txt    425_reversi_game.txt
23774: 240_unitGlobal_tester.txt   426_IBUtils.TXT
23775: 241_db3_sql_tutorial.txt    427_IBDatabase.TXT
23776: 241_db3_sql_tutorial2.txt   428_SortGrid.TXT
23777: 241_db3_sql_tutorial2fix.txt 429_fileclass.txt
23778: 241_db3_sql_tutorial3.txt   430_fileoperation.txt
23779: 241_db3_sql_tutorial3connect.txt 430_fileoperation_tester.txt
23780: 241_db3_sql_tutorial3_fpstest.txt 431_performance_index.txt
23781: 241_RTL_SET2.txt           432_shortstring_routines.txt
23782: 241_RTL_SET2_tester.txt    433_video_avicap.txt
23783: 242_Component_Control.txt  433_video_avicap2.txt
23784: 243_tutorial_loader.txt    434_GSM_module.TXT
23785: 244_script_loader_loop.txt  435_httpcommon.txt
23786: 245_formapp2.txt           436_GraphicSplitter.txt
23787: 245_formapp2_tester.txt    436_GraphicSplitter_form.txt
23788: 245_formapp2_testerX.txt   436_GraphicSplitter_form2.txt
23789: 246_httpapp.txt            436_teetest_screen.TXT
23790: 247_datecalendar.txt       436_teetest_screen2.TXT
23791: 248_ASCII_Grid2_sorted.TXT 437_WinAPItop.txt
23792: 249_picture_grid.TXT      437_WinAPItop_Firebirdtester.txt
23793: 250_tipsandtricks2.txt    438_OvcInternational.txt
23794: 250_tipsandtricks3.txt    439_AsyncFreeDemo.txt
23795: 250_tipsandtricks3api.txt 439_AsyncFreeDemoForm.txt

```

```

23796: 250_tipsandtricks3_admin_elevation.txt          440_DLL_Tutor.txt
23797: 250_tipsandtricks3_tester.txt                 440_DLL_Tutor2.txt
23798: 250_tipsandtricks4_tester.txt                 440_XML_Tutor.txt
23799: 250_tipsandtricks4_tester2.txt                440_XML_Tutor2.txt
23800: 251_compare_noise_gauss.txt                  441_make_app.txt
23801: 251_whitenoise.txt                          442_arduino_rgb_led.txt
23802: 251_whitenoise2.txt                         443_webserver_arduino_rgb_light.txt
23803: 252_hilbert_turtle.txt                      443_webserver_arduino_rgb_light4.txt
23804: 252_pas_hilbert.txt                        444_webserver_arduino3ibz_rgb_led_basta.txt
23805: 253_opearatingsystem3.txt                  445_datagrid.txt
23806: 254_dynarrays.txt                         445_datagrid2.txt
23807: 255_einstein.txt                           445_datagrid_android_arduino.txt
23808: 256_findconsts_of_EXE.txt                 446_arduino_timer.txt
23809: 256_findfunctions2_of_EXE.txt              447_patternFrm_mx3.txt
23810: 256_findfunctions2_of_EXEaverp.txt        448_Synapse.txt
23811: 256_findfunctions2_of_EXEspec.txt         448_Synapse2.txt
23812: 256_findfunctions3.txt                     449_dweb_start_tester.txt
23813: 256_findfunctions_of_EXE.txt               450_Synapse_HTTPS.txt
23814: 257_AES_Cipher.txt                         450_Synapse_Mime.txt
23815: 258_AES_cryptobox.txt                     450_Synapse_ScanPing.txt
23816: 258_AES_cryptobox2.txt                    451_ocx_player.txt
23817: 258_AES_cryptobox2_passdlg.txt           451_OCX_Winplayer2.txt
23818: 259_AES_crypt_directory.txt              452_dbtreeview.txt
23819: 260_sendmessage_2.TXT                   452_dbtreeview2.txt
23820: 260_sendmessage_beta.TXT                453_stdfuncs.txt
23821: 261_probability.txt                      454_fileStream.txt
23822: 262_mxoutputdemo4.txt                   455_functionfun.txt
23823: 263_async_sound.txt                     455_functionfun2.txt
23824: 264_vclutils.txt                        455_functionfun2_test.txt
23825: 264_VCL_utils2.txt                      457_ressource_grid.txt
23826: 265_timer_API.txt                       458_atomimageX.txt
23827: 266_serial_interface.txt               459_cindyfunc.txt
23828: 266_serial_interface2.txt              459_cindyfunc2.txt
23829: 266_serial_interface3.txt              460_TopTenFunctions.txt
23830: 267_ackermann_rec.txt                 461_sqIFORM_calwin.txt
23831: 267_ackermann_variants.txt            462_caesarcipher.txt
23832: 268_DBGrid_tree.txt                    463_global_exception.txt
23833: 269_record_grid.TXT                  464_function_procedure.txt
23834: 270_Jedi_FunctionPower.txt            464_function_procedure2.txt
23835: 270_Jedi_FunctionPowertester.txt       464_function_procedure3.txt
23836: 271_closures_study.txt                465_U_HexView.txt
23837: 271_closures_study_workingset2.txt    466_moon.txt
23838: 272_pas_function_show.txt             466_moon_inputquery.txt
23839: 273_pas_function_show2.txt            4671_cardmagic.txt
23840: 274_library_functions.txt            467_helmholtz_graphic.txt
23841: 275_turtle_language.txt              468_URLMon.txt
23842: 275_turtle_language_save.txt         468_URLMon2.txt
23843: 276_save_algo.txt                   469_formarrow.txt
23844: 276_save_algo2.txt                 469_formarrow_datepicker.txt
23845: 277_functionsfor39.txt            469_formarrow_datepicker_ibz_result.txt
23846: 278_DB_Dialogs.TXT                 469_ibzresult.txt
23847: 279_hexer2.TXT                   470_DFFUtils_compiled.txt
23848: 279_hexer2macro.TXT              470_DFFUtils_ScrollingLED.txt
23849: 279_hexer2macroback.TXT           470_Oscilloscope.txt
23850: 280_UML_process.txt              470_Oscilloscope_code.txt
23851: 280_UML_process_knabe2.txt        471_cardmagic.txt
23852: 280_UML_process_knabe3.txt        471_cardmagic2.txt
23853: 280_UML_process_TIM_Botzenhardt.txt 472_allcards.txt
23854: 280_UML_TIM_Seitz.txt            473_comboset.txt
23855: 281_picturepuzzle.txt            474_wakeonlan.txt
23856: 281_picturepuzzle2.txt           474_wakeonlan2.txt
23857: 281_picturepuzzle3.txt           476_getscripttest.txt
23858: 281_picturepuzzle4.txt           477_filenameonly.txt
23859: 479_inputquery.txt               480_regex_pathfinder.txt
23860: 480_regex_pathfinder2.txt         481_processList.txt
23861: 482_processPipe.txt              482_processPipeGCC.txt
23862: 483_PathFuncTest_mx.txt          484_filefinder3.txt
23863: 485_InnoFunc.txt                486_VideoGrabber.txt
23864: 487_asyncKeyState.txt           488_asyncTerminal.txt
23865: 489_simpleComport.txt          490_webCamproc.txt
23866: 491_analogmeter.txt            492_snowflake2.txt
23867: 493_gadgets.txt                495_fourierfreq.txt
23868: 496_InstallX.txt              497_LED.txt
23869: 498_UnitTesting.txt            499_mulu42.txt
23870: 500_diceoflifes.txt            501_firebird_datasnap_tests.txt
23871: 502_findalldocs.txt           503_led_switch.txt
23872: 504_fileclass.txt             505_debug.txt
23873: 506_colormatrix.txt           507_derutils.txt
23874: 508_simplecomportmorse.txt     509_GEOMap2.txt
23875: 509_509_GEOMap2_SReverse.TXT   510_510_bonn_gpsdata_mx4.pas
23876: 511_LEDLabel.txt              512_LED_moon.txt
23877: 513_StreamIntegration.txt      514_LED_moon2.txt
23878: 515_lcdclock3.txt             516_mapview.txt
23879: 517_animation7.txt            518_sensors_meter.txt
23880: 519_powtills.txt              520_run bytecode.txt
23881: 521_iputils2.txt              522_getgeocode.txt
23882: 523_NMEA.txt                 524_NAV_Utils.txt
23883: 525_GEO84s.txt                526_Compass_meter.txt
23884: 527_GPSDemo.txt              528_linescount.txt

```

```

23885: 529_profilertest.txt
23886: 531_profilertest.txt
23887: 533_syncasync_demo.txt
23888: 535_Battleship3.pas
23889: 537_iniplus.TXT
23890: 539_timeturtle123.txt
23891: 541_webserver_arduino_motorturtle.txt
23892: 543_MATH_TurboP.PAS
23893: 545_strips.TXT
23894: 547_regexmaster.TXT
23895: 549_3D_Panorama.txt
23896: 551_ArduinoTester.txt
23897: 553_ArduinoCockBit3.txt
23898: 556_stringlistrandom.TXT
23899: 558_highrestimer.TXT - 559_highthrestimer2
23900: 561_newfunctions399160.txt
23901: 563_moonpaper.txt
23902: 565_ConsoleCapture.txt
23903: 567_SquareWordGrids2.txt
23904: 569_keylog.txt - 569_ServiceMgr2.TXT
23905: 571_myPing.txt
23906: 573_modbusfrm_Main.pas
23907: 575_TARTARUGA/Desktop.txt
23908: 577_listbox2list.txt
23909: 579_numbersystems_sort.txt
23910: 581_stringstream.txt
23911: 583_VirtualConstructor_savereport.txt
23912: 585_fulltextfinder_cleancode_override.txt
23913: 587_one_function.txt
23914: 589_avi_animate.txt
23915: 591_emailattach.txt
23916: 593_round_time.txt
23917: 595_check_memory.txt
23918: 597_ole_commands.txt
23919: 599_bug.txt
23920:
23921: ( 589_AVI_Throbber.res.txt) 589_avi_animate.txt
23922:
23923: http://sourceforge.net/projects/maxbox/files/Examples/
23924:
23925: Help Online:
23926: http://max.kleiner.com/maxbox_functions_all.htm
23927:
23928: WebScript Examples:
23929:
23930: http://www.softwareschule.ch/examples/performer.txt;
23931: http://www.softwareschule.ch/examples/turtle.txt;
23932: http://www.softwareschule.ch/examples/SQLExport.txt;
23933: http://www.softwareschule.ch/examples/Richter.txt;
23934: http://www.softwareschule.ch/examples/checker.txt;
23935: http://www.softwareschule.ch/examples/demoscript.txt;
23936: http://www.softwareschule.ch/examples/ibzresult.txt;
23937: http://www.softwareschule.ch/examples/performindex.txt
23938: http://www.softwareschule.ch/examples/processlist.txt
23939: http://www.softwareschule.ch/examples/game.txt
23940: http://www.softwareschule.ch/examples/GEOGPS.txt
23941: http://www.softwareschule.ch/examples/turtle2.txt
23942: http://www.softwareschule.ch/examples/turtle3.txt
23943: http://www.softwareschule.ch/examples/asyncterminal.txt
23944: http://www.softwareschule.ch/examples/snowflake.txt
23945: http://www.softwareschule.ch/examples/arduinooled.txt
23946: http://www.softwareschule.ch/examples/moon2.txt
23947: http://www.softwareschule.ch/examples/cockpit.txt
23948: http://www.softwareschule.ch/examples/tartaruga.txt
23949: http://www.softwareschule.ch/examples/surprise.txt
23950:
23951:
23952: Delphi Basics Run Time Library listing
23953: ****
23954: A
23955: Compiler Directive $A Determines whether data is aligned or packed
23956: Compiler Directive $Align Determines whether data is aligned or packed
23957: Compiler Directive $AppType Determines the application type : GUI or Console
23958: Procedure SysUtils Abort Aborts the current processing with a silent exception
23959: Function System Abs Gives the absolute value of a number (-ve sign is removed)
23960: Directive Abstract Defines a class method only implemented in subclasses
23961: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
23962: Function System Addr Gives the address of a variable, function or procedure
23963: Keyword And Boolean and or bitwise and of two arguments
23964: Type System AnsiChar A character type guaranteed to be 8 bits in size
23965: Function SysUtils AnsiCompareStr Compare two strings for equality
23966: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
23967: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
23968: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
23969: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
23970: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
23971: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
23972: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
23973: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string

```

23974: **Function** StrUtils AnsiPos Find the position **of** one **string** in another
23975: **Function** StrUtils AnsiReplaceStr Replaces a part **of** one **string** with another
23976: **Function** StrUtils AnsiReverseString Reverses the sequence **of** letters **in** a **string**
23977: **Function** StrUtils AnsiRightStr Extracts characters from the right **of** a **string**
23978: **Function** StrUtils AnsiStartsStr Returns true **if** a **string** starts **with** a substring
23979: **Type** System AnsiString A data **type** that holds a **string** **of** AnsiChars
23980: **Function** SysUtils AnsiUpperCase Change lower **case** characters **in** a **string** **to** upper **case**
23981: **Procedure** System Append Open a text **file** **to** allow appending **of** text **to** the **end**
23982: **Procedure** SysUtils AppendStr Concatenate one **string** **onto** the **end** **of** another
23983: **Function** Math ArcCos The Arc Cosine **of** a number, returned **in** radians
23984: **Function** Math ArcSin The Arc Sine **of** a number, returned **in** radians
23985: **Function** System ArcTan The Arc Tangent **of** a number, returned **in** radians
23986: **Keyword** **Array** A data **type** holding indexable collections **of** data
23987: **Keyword** **As** Used **for** casting **object** references
23988: **Procedure** System Assign Assigns a **file** handle **to** a binary or text **file**
23989: **Function** System Assigned Returns true **if** a reference **is** not nil
23990: **Procedure** System AssignFile Assigns a **file** handle **to** a binary or text **file**
23991: **Procedure** Printers AssignPrn Treats the printer **as** a text **file** - an easy way **of** printing text
23992:
23993: B
23994: Compiler Directive \$B Whether **to** short cut **and** **and** **or** operations
23995: Compiler Directive \$BoolEval Whether **to** short cut **and** **and** **or** operations
23996: **Procedure** SysUtils Beep Make a beep sound
23997: **Keyword** **Begin** Keyword that starts a statement block
23998: **Function** System BeginThread Begins a separate thread **of** code execution
23999: **Procedure** System BlockRead Reads a block **of** data records from an untyped binary **file**
24000: **Procedure** System BlockWrite Writes a block **of** data records **to** an untyped binary **file**
24001: **Type** System Boolean Allows just True **and** False values
24002: **Function** Classes Bounds Create a TRect value from top left **and** size values
24003: **Procedure** System Break Forces a jump **out** **of** a single loop
24004: **Type** System Byte An integer **type** supporting values 0 **to** 255
24005:
24006: C
24007: **Type** System Cardinal The basic unsigned integer **type**
24008: **Keyword** **Case** A mechanism **for** acting upon different values **of** an Ordinal
24009: **Function** StdConv CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
24010: **Function** SysUtils ChangeFileExt Change the extension part **of** a **file** name
24011: **Type** System Char Variable **type** holding a single character
24012: **Procedure** System ChDir Change the working drive plus path **for** a specified drive
24013: **Function** System Chr Convert an integer into a character
24014: **Keyword** **Class** Starts the declaration **of** a **type** **of** object **class**
24015: **Procedure** System Close Closes an open **file**
24016: **Procedure** System CloseFile Closes an open **file**
24017: **Variable** System CmdLine Holds the execution text used **to** start the current **program**
24018: **Type** System Comp A 64 bit signed integer
24019: **Function** SysUtils CompareStr Compare two strings **to** see which **is** greater than the other
24020: **Function** SysUtils CompareText Compare two strings **for** equality, ignoring **case**
24021: **Function** Math CompareValue Compare numeric values **with** a tolerance
24022: **Function** System Concat Concatenates one **or** more strings **into** one **string**
24023: **Keyword** **Const** Starts the definition **of** fixed data values
24024: **Keyword** **Constructor** Defines the method used **to** create an **object** **from** a **class**
24025: **Procedure** System Continue Forces a jump **to** the next iteration **of** a loop
24026: **Function** ConvUtils Convert Convert one measurement value **to** another
24027: **Function** System Copy Create a copy **of** part **of** a **string** **or** an **array**
24028: **Function** System Cos The Cosine **of** a number
24029: **Function** SysUtils CreateDir Create a directory
24030: **Type** System Currency A floating point **type** **with** 4 decimals used **for** financial values
24031: **Variable** SysUtils CurrencyDecimals Defines decimal digit count **in** the Format **function**
24032: **Variable** SysUtils CurrencyFormat Defines currency **string** placement **in** curr display functions
24033: **Variable** SysUtils CurrencyString The currency **string** used **in** currency display functions
24034: **Function** SysUtils CurrToStr Convert a currency value **to** a **string**
24035: **Function** SysUtils CurrToStrF Convert a currency value **to** a **string** **with** formatting
24036:
24037: D
24038: Compiler Directive \$D Determines whether application debug information **is** built
24039: Compiler Directive \$DebugInfo Determines whether application debug information **is** built
24040: Compiler Directive \$Define Defines a compiler directive symbol - **as** used by IfDef
24041: Compiler Directive \$DefinitionInfo Determines whether application symbol information **is** built
24042: **Function** SysUtils Date Gives the current date
24043: **Variable** SysUtils DateSeparator The character used **to** separate display date fields
24044: **Function** SysUtils DateTimeToFileDate Convert a TDateTime value **to** a File date/time format
24045: **Function** SysUtils DateTimeToStr Converts TDateTime date **and** time values **to** a **string**
24046: **Procedure** SysUtils DateTimeToString Rich formatting **of** a TDateTime variable **into** a **string**
24047: **Function** SysUtils DateToStr Converts a TDateTime date value **to** a **string**
24048: **Function** DateUtils DayOfTheMonth Gives day **of** month index **for** a TDateTime value (ISO 8601)
24049: **Function** DateUtils DayOfTheWeek Gives day **of** week index **for** a TDateTime value (ISO 8601)
24050: **Function** DateUtils DayOfTheYear Gives the day **of** the year **for** a TDateTime value (ISO 8601)
24051: **Function** SysUtils DayOfWeek Gives day **of** week index **for** a TDateTime value
24052: **Function** DateUtils DaysBetween Gives the whole number **of** days between 2 dates
24053: **Function** DateUtils DaysInAMonth Gives the number **of** days **in** a month
24054: **Function** DateUtils DaysInAYear Gives the number **of** days **in** a year
24055: **Function** DateUtils DaySpan Gives the fractional number **of** days between 2 dates
24056: **Procedure** System Dec Decrement an ordinal variable
24057: **Variable** SysUtils DecimalSeparator The character used **to** display the decimal point
24058: **Procedure** SysUtils DecodeDate Extracts the year, month, day values **from** a TDateTime **var**.
24059: **Procedure** DateUtils DecodeDateTime Breaks a TDateTime variable **into** its date/time parts
24060: **Procedure** SysUtils DecodeTime Break a TDateTime value **into** individual time values
24061: **Directive** **Default** Defines default processing **for** a **property**
24062: **Function** Math DegToRad Convert a degrees value **to** radians

24063: **Procedure** System Delete Delete a section **of** characters from a **string**
 24064: **Function** SysUtils DeleteFile Delete a **file** specified by its **file name**
 24065: **Keyword Destructor** Defines the method used **to** destroy an **object**
 24066: **Function** SysUtils DirectoryExists Returns true **if** the given directory exists
 24067: **Function** SysUtils DiskFree Gives the number **of** free bytes **on** a specified drive
 24068: **Function** SysUtils DiskSize Gives the size **in bytes of** a specified drive
 24069: **Procedure** System Dispose Dispose **of** storage used by a pointer **type** variable
 24070: **Keyword Div** Performs integer division, discarding the remainder
 24071: **Keyword Do** Defines the start **of** some controlled action
 24072: **Type** System Double A floating point **type** supporting about 15 digits **of** precision
 24073: **Keyword DownTo** Prefixes an decremental **for** loop target value
 24074: **Function** StrUtils DupeString Creates a **string** containing copies **of** a substring
 24075: **Directive Dynamic** Allows a **class** method **to** be overriden **in** derived classes
 24076:
 24077: E
 24078: Compiler Directive \$Else Starts the alternate section **of** an IfDef **or** IfNDef
 24079: Compiler Directive \$EndIf Terminates conditional code compilation
 24080: Compiler Directive \$ExtendedSyntax Controls some **Pascal** extension handling
 24081: **Keyword Else** Starts false section **of if, case and try statements**
 24082: **Function** SysUtils EncodeDate Build a TDateTime value from year, month **and** day values
 24083: **Function** DateUtils EncodeDateTime Build a TDateTime value from day **and** time values
 24084: **Function** SysUtils EncodeTime Build a TDateTime value from hour, min, sec **and** msec values
 24085: **Keyword End** Keyword that terminates statement blocks
 24086: **Function** DateUtils EndOfDay Generate a TDateTime value **set to** the very **end of** a day
 24087: **Function** DateUtils EndOfMonth Generate a TDateTime value **set to** the very **end of** a month
 24088: **Procedure** System EndThread Terminates a thread **with** an exit code
 24089: **Function** System EOF Returns true **if** a **file** opened **with** Reset **is** at the **end**
 24090: **Function** System Eoln Returns true **if** the current text **file is** pointing at a line **end**
 24091: **Procedure** System Erase Erase a **file**
 24092: **Variable** System ErrorAddr Sets the error address when an application terminates
 24093: **Keyword Except** Starts the error trapping clause **of a Try statement**
 24094: **Procedure** System Exclude Exclude a value **in a set** variable
 24095: **Procedure** System Exit Exit abruptly **from a function or procedure**
 24096: **Variable** System ExitCode Sets the return code when an application terminates
 24097: **Function** System Exp Gives the exponent **of** a number
 24098: **Directive System Export** Makes a **function or procedure** **in** a DLL externally available
 24099: **Type** System Extended The floating point **type** with the highest capacity **and** precision
 24100: **Function** SysUtils ExtractFileDir Extracts the dir part **of** a full **file name**
 24101: **Function** SysUtils ExtractFileDrive Extracts the drive part **of** a full **file name**
 24102: **Function** SysUtils ExtractFileExt Extracts the extension part **of** a full **file name**
 24103: **Function** SysUtils ExtractFileName Extracts the name part **of** a full **file name**
 24104: **Function** SysUtils ExtractFilePath Extracts the path part **of** a full **file name**
 24105:
 24106: F
 24107: **Function** StdConv FahrenheitToCelsius Convert a fahrenheit temperature into celsius
 24108: **Keyword File** Defines a typed **or** untyped **file**
 24109: **Function** SysUtils FileAge Get the last modified date/time **of a file** without opening it
 24110: **Function** SysUtils FileDateToDate Time Converts a **file** date/time format **to** a TDateTime value
 24111: **Function** SysUtils FileExists Returns true **if** the given **file** exists
 24112: **Function** SysUtils FileGetAttr Gets the attributes **of a file**
 24113: **Variable** System FileMode Defines how Reset opens a binary **file**
 24114: **Function** System FilePos Gives the **file position** **in** a binary **or** text **file**
 24115: **Function** SysUtils FileSearch Search **for a file** **in** one **or** more directories
 24116: **Function** SysUtils FileSetAttr Sets the attributes **of a file**
 24117: **Function** SysUtils FileSetDate Set the last modified date **and** time **of a file**
 24118: **Function** System FileInfo Gives the size **in records** **of an open file**
 24119: **Procedure** System FillChar Fills **out** a section **of** storage **with** a fill character **or** byte value
 24120: **Keyword Finally** Starts the unconditional code section **of a Try statement**
 24121: **Function** SysUtils FindClose Closes a successful FindFirst **file search**
 24122: **Function** SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
 24123: **Function** SysUtils FindFirst Finds all files matching a **file mask and attributes**
 24124: **Function** SysUtils FindNext Find the next **file** after a successful FindFirst
 24125: **Function** SysUtils FloatToStr Convert a floating point value **to a string**
 24126: **Function** SysUtils FloatToStrF Convert a floating point value **to a string with** formatting
 24127: **Procedure** System Flush Flushes buffered text **file** data **to the file**
 24128: **Keyword For** Starts a loop that executes a finite number **of times**
 24129: **Function** SysUtils ForceDirectories Create a new path **of** directories
 24130: **Function** SysUtils Format Rich formatting **of numbers and text** **into a string**
 24131: **Function** SysUtils FormatCurr Rich formatting **of a currency value** **into a string**
 24132: **Function** SysUtils FormatDateTime Rich formatting **of a TDateTime variable** **into a string**
 24133: **Function** SysUtils FormatFloat Rich formatting **of a floating point number** **into a string**
 24134: **Function** System Frac The fractional part **of** a floating point number
 24135: **Procedure** SysUtils FreeAndNil Free memory **for an object and set it to nil**
 24136: **Procedure** System FreeMem Free memory storage used by a variable
 24137: **Keyword System Function** Defines a subroutine that returns a value
 24138:
 24139: G
 24140: **Function** SysUtils GetCurrentDir Get the current directory (drive plus directory)
 24141: **Procedure** System GetDir Get the **default** directory (drive plus path) **for** a specified drive
 24142: **Function** System GetLastError Gives the error code **of** the last failing Windows API call
 24143: **Procedure** SysUtils GetLocaleFormatSettings Gets locale values **for** thread-safe functions
 24144: **Function** System GetMem Get a specified number **of** storage bytes
 24145: **Keyword Goto** Forces a jump **to a label**, regardless **of nesting**
 24146:
 24147: H
 24148: Compiler Directive \$H Treat **string** types **as** AnsiString **or** ShortString
 24149: Compiler Directive \$Hints Determines whether Delphi shows compilation hints
 24150: **Procedure** System Halt Terminates the **program with** an optional dialog
 24151: **Function** System Hi Returns the hi-order byte **of a (2 byte)** Integer

24152: **Function** System High Returns the highest value of a type or variable
 24153:
 24154: I
 24155: Compiler Directive \$I Allows code in an include file to be incorporated into a Unit
 24156: Compiler Directive \$IfDef Executes code if a conditional symbol has been defined
 24157: Compiler Directive \$IfNDef Executes code if a conditional symbol has not been defined
 24158: Compiler Directive \$IfOpt Tests for the state of a Compiler directive
 24159: Compiler Directive \$Include Allows code in an include file to be incorporated into a Unit
 24160: Compiler Directive \$IOChecks When on, an IO operation error throws an exception
 24161: Keyword If Starts a conditional expression to determine what to do next
 24162: Keyword Implementation Starts the implementation (code) section of a Unit
 24163: Keyword In Used to test if a value is a member of a set
 24164: **Procedure** System Inc Increment an ordinal variable
 24165: **Function** DateUtils IncDay Increments a TDateTime variable by + or - number of days
 24166: **Procedure** System Include Include a value in a set variable
 24167: **Function** DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
 24168: **Function** DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
 24169: **Function** SysUtils IncMonth Increments a TDateTime variable by a number of months
 24170: **Function** DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
 24171: **Function** DateUtils IncYear Increments a TDateTime variable by a number of years
 24172: Directive Index Principally defines indexed class data properties
 24173: Constant Math Infinity Floating point value of infinite size
 24174: Keyword Inherited Used to call the parent class constructor or destructor method
 24175: Variable System Input Defines the standard input text file
 24176: **Function** Dialogs InputBox Display a dialog that asks for user text input, with default
 24177: **Function** Dialogs InputQuery Display a dialog that asks for user text input
 24178: **Procedure** System Insert Insert a string into another string
 24179: **Function** System Int The integer part of a floating point number as a float
 24180: Type System Int64 A 64 bit sized integer - the largest in Delphi
 24181: Type System Integer The basic Integer type
 24182: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
 24183: **Function** SysUtils IntToHex Convert an Integer into a hexadecimal string
 24184: **Function** SysUtils IntToStr Convert an integer into a string
 24185: **Function** System IOResult Holds the return code of the last I/O operation
 24186: Keyword Is Tests whether an object is a certain class or descendant
 24187: **Function** Math IsInfinite Checks whether a floating point number is infinite
 24188: **Function** SysUtils IsLeapYear Returns true if a given calendar year is a leap year
 24189: **Function** System IsMultiThread Returns true if the code is running multiple threads
 24190: **Function** Math IsNaN Checks to see if a floating point number holds a real number
 24191:
 24192: L
 24193: Compiler Directive \$L Determines what application debug information is built
 24194: Compiler Directive \$LocalSymbols Determines what application debug information is built
 24195: Compiler Directive \$LongStrings Treat string types as AnsiString or ShortString
 24196: **Function** SysUtils LastDelimiter Find the last position of selected characters in a string
 24197: **Function** System Length Return the number of elements in an array or string
 24198: **Function** System Ln Gives the natural logarithm of a number
 24199: **Function** System Lo Returns the low-order byte of a (2 byte) Integer
 24200: **Function** Math Log10 Gives the log to base 10 of a number
 24201: Variable SysUtils LongDateFormat Long version of the date to string format
 24202: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
 24203: Type System LongInt An Integer whose size is guaranteed to be 32 bits
 24204: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
 24205: Variable SysUtils LongTimeFormat Long version of the time to string format
 24206: Type System LongWord A 32 bit unsigned integer
 24207: **Function** System Low Returns the lowest value of a type or variable
 24208: **Function** SysUtils LowerCase Change upper case characters in a string to lower case
 24209:
 24210: M
 24211: Compiler Directive \$MinEnumSize Sets the minimum storage used to hold enumerated types
 24212: **Function** Math Max Gives the maximum of two integer values
 24213: Constant System MaxInt The maximum value an Integer can have
 24214: Constant System MaxLongInt The maximum value an LongInt can have
 24215: **Function** Math Mean Gives the average for a set of numbers
 24216: **Function** Dialogs MessageDlg Displays a message, symbol, and selectable buttons
 24217: **Function** Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
 24218: **Function** Math Min Gives the minimum of two integer values
 24219: Constant SysUtils MinsPerDay Gives the number of minutes in a day
 24220: **Procedure** System MkDir Make a directory
 24221: Keyword Mod Performs integer division, returning the remainder
 24222: Constant SysUtils MonthDays Gives the number of days in a month
 24223: **Function** DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
 24224: **Procedure** System Move Copy bytes of data from a source to a destination
 24225:
 24226: N
 24227: Constant Math NaN Not a real number
 24228: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
 24229: **Procedure** System New Create a new pointer type variable
 24230: Constant System Nil A pointer value that is defined as undetermined
 24231: Keyword Not Boolean Not or bitwise not of one arguments
 24232: **Function** SysUtils Now Gives the current date and time
 24233: Variable Variants Null A variable that has no value
 24234:
 24235: O
 24236: Compiler Directive \$O Determines whether Delphi optimises code when compiling
 24237: Compiler Directive \$Optimization Determines whether Delphi optimises code when compiling
 24238: Compiler Directive \$OverflowChecks Determines whether Delphi checks integer and enum bounds
 24239: Keyword System Object Allows a subroutine data type to refer to an object method
 24240: **Function** System Odd Tests whether an integer has an odd value

24241: Keyword **Of** Linking keyword used **in** many places
24242: Keyword **On** Defines exception handling **in a Try Except clause**
24243: Keyword **Or** Boolean **or** **or** bitwise **or** **of** two arguments
24244: **Function** System Ord Provides the Ordinal value **of** an integer, character **or** enum
24245: Directive **Out** Identifies a routine parameter **for** output only
24246: Variable System Output Defines the standard output text **file**
24247: Directive **Overload** Allows **2 or more** routines **to** have the same name
24248: Directive **Override** Defines a method that replaces a **virtual** parent **class** method
24249:
24250: **P**
24251: Keyword **Packed** Compacts complex data types into minimal storage
24252: **Type** System PAnsiChar A pointer **to** an AnsiChar value
24253: **Type** System PAnsiString Pointer **to** an AnsiString value
24254: **Function** System ParamCount Gives the number **of** parameters passed **to** the current **program**
24255: **Function** System ParamStr Returns one **of** the parameters used **to** run the current **program**
24256: **Type** System PChar A pointer **to** an Char value
24257: **Type** System PCurrency Pointer **to** a Currency value
24258: **Type** System PDateTime Pointer **to** a TDateTime value
24259: **Type** System PExtended Pointer **to** a Extended floating point value
24260: **Function** System Pi The mathematical constant
24261: **Type** System PInt64 Pointer **to** an Int64 value
24262: **Function** Classes Point Generates a TPoint value from X **and** Y values
24263: **Type** System Pointer Defines a general use Pointer **to** any memory based data
24264: **Function** Classes PointsEqual Compares two TPoint values **for equality**
24265: **Function** System Pos Find the position **of** one **string** **in** another
24266: **Function** System Pred Decrement an ordinal variable
24267: **Function** Printers Printer Returns a reference **to** the global Printer **object**
24268: Directive **Private** Starts the section **of private** data **and** methods **in a class**
24269: Keyword System **Procedure** Defines a subroutine that does **not** return a value
24270: **Procedure** FileCtrl ProcessPath Split a drive/path/filename **string** into its constituent parts
24271: Keyword System **Program** Defines the start **of** an application
24272: **Function** Dialogs PromptForFileName Shows a dialog allowing the user **to** select a **file**
24273: Keyword System **Property** Defines controlled access **to class** fields
24274: Directive **Protected** Starts a section **of class private** data accessible **to** sub-classes
24275: **Type** System PShortString A pointer **to** an ShortString value
24276: **Type** System PString Pointer **to** a String value
24277: **Function** Types PtInRect Tests **to see if** a point lies within a rectangle
24278: Directive **Public** Starts an externally accessible section **of a class**
24279: Directive **Published** Starts a published externally accessible section **of a class**
24280: **Type** System PVariant Pointer **to** a Variant value
24281: **Type** System PWideChar Pointer **to** a WideChar
24282: **Type** System PWideString Pointer **to** a WideString value
24283:
24284: **Q**
24285: Compiler Directive **\$Q** Determines whether Delphi checks integer **and** enum bounds
24286:
24287: **R**
24288: Compiler Directive **\$R** Determines whether Delphi checks array bounds
24289: Compiler Directive **\$RangeChecks** Determines whether Delphi checks array bounds
24290: Compiler Directive **\$ReferenceInfo** Determines whether symbol reference information **is built**
24291: Compiler Directive **\$Resource** Defines a resource **file** **to** be included **in** the application linking
24292: **Function** Math RadToDeg Converts a radian value **to** degrees
24293: Keyword **Raise** Raise an exception
24294: **Function** System Random Generate a random floating point **or** integer number
24295: **Procedure** System Randomize Reposition the Random number generator next value
24296: **Function** Math RandomRange Generate a random integer number within a supplied range
24297: Variable System RandSeed Reposition the Random number generator next value
24298: **Procedure** System Read Read data from a binary **or** text **file**
24299: **Procedure** System ReadLn Read a complete line **of** data from a text **file**
24300: **Type** System Real A floating point **type** supporting about **15** digits **of** precision
24301: **Type** System Real48 The floating point **type** **with** the highest capacity **and** precision
24302: **Procedure** System ReallocMem Reallocate an existing block **of** storage
24303: **Function** DateUtils RecodeDate Change only the date part **of** a TDateTime variable
24304: **Function** DateUtils RecodeTime Change only the time part **of** a TDateTime variable
24305: Keyword **Record** A structured data **type** - holding fields **of** data
24306: **Function** Classes Rect Create a TRect value from **2** points **or** **4** coordinates
24307: **Function** SysUtils RemoveDir Remove a directory
24308: **Procedure** System Rename Rename a **file**
24309: **Function** SysUtils RenameFile Rename a **file** **or** directory
24310: Keyword **Repeat** Repeat statements **until** a termination condition **is met**
24311: **Procedure** SysUtils ReplaceDate Change only the date part **of** a TDateTime variable
24312: **Procedure** SysUtils ReplaceTime Change only the time part **of** a TDateTime variable
24313: **Procedure** System Reset Open a text **file** **for reading**, **or** binary **file** **for read/write**
24314: Variable System Result A variable used **to** hold the return value from a **function**
24315: **Procedure** System ReWrite Open a text **or** binary **file** **for write** access
24316: **Procedure** System RmDir Remove a directory
24317: **Function** System Round Rounds a floating point number **to** an integer
24318: **Procedure** System RunError Terminates the **program** **with** an error dialog
24319:
24320: **S**
24321: Constant SysUtils SecsPerDay Gives the number **of** seconds **in** a day
24322: **Procedure** System Seek Move the pointer **in** a binary **file** **to** a new **record** position
24323: **Function** System SeekEOF Skip **to** the end **of** the current line **or** **file**
24324: **Function** System SeekEOLN Skip **to** the end **of** the current line **or** **file**
24325: **Function** FileCtrl SelectDirectory Display a dialog **to** allow user selection **of** a directory
24326: Variable System Self Hidden parameter **to** a method - refers **to** the containing **object**
24327: Keyword **Set** Defines a **set** **of** up **to** **255** distinct values
24328: **Function** SysUtils SetCurrentDir Change the current directory
24329: **Procedure** System SetLength Changes the size **of** a **string**, **or** the size(s) **of** an **array**

24330: **Procedure** System SetString Copies characters from a buffer into a **string**
 24331: Keyword **Shl** Shift an integer value left by a number **of** bits
 24332: Variable SysUtils ShortDateFormat Compact version **of** the date **to** **string** format
 24333: Variable SysUtils ShortDayNames An **array** **of** days **of** the week names, starting **1 = Sunday**
 24334: **Type** System ShortInt An integer **type** supporting values **-128 to 127**
 24335: Variable SysUtils ShortMonthNames An **array** **of** days **of** the month names, starting **1 = Jan**
 24336: **Type** System **ShortString** Defines a **string** **of** up **to** **255** characters
 24337: Variable SysUtils ShortTimeFormat Short version **of** the time **to** **string** format
 24338: **Procedure** Dialogs ShowMessage Display a **string** **in** a simple dialog **with** an OK button
 24339: **Procedure** Dialogs ShowMessageFmt Display formatted data **in** a simple dialog **with** an OK button
 24340: **Procedure** Dialogs ShowMessagePos Display a **string** **in** a simple dialog at a given screen position
 24341: Keyword **Shr** Shift an integer value right by a number **of** bits
 24342: **Function** System Sin The Sine **of** a number
 24343: **Type** System Single The smallest capacity **and** precision floating point **type**
 24344: **Function** System SizeOf Gives the storage byte size **of** a **type** **or** variable
 24345: **Function** System Slice Creates a slice **of** an **array** **as** an Open **Array** parameter
 24346: **Type** System SmallInt An Integer **type** supporting values from **-32768 to 32767**
 24347: **Function** System Sqr Gives the square **of** a number
 24348: **Function** System Sqrt Gives the square root **of** a number
 24349: **Procedure** System Str Converts an integer **or** floating point number **to** a **string**
 24350: **Type** System **String** A data **type** that holds a **string** **of** characters
 24351: **Function** System StringOfChar Creates a **string** **with** one character repeated many times
 24352: **Function** SysUtils StringReplace Replace one **or** more substrings found within a **string**
 24353: **Function** System StringToWideChar Converts a normal **string** **into** a WideChar **0** terminated buffer
 24354: **Function** SysUtils StrScan Searches **for** a specific character **in** a constant **string**
 24355: **Function** SysUtils StrToCurr Convert a number **string** **into** a currency value
 24356: **Function** SysUtils StrToDate Converts a date **string** **into** a TDateTime value
 24357: **Function** SysUtils StrToDateTime Converts a date+time **string** **into** a TDateTime value
 24358: **Function** SysUtils StrToFloat Convert a number **string** **into** a floating point value
 24359: **Function** SysUtils StrToInt Convert an integer **string** **into** an Integer value
 24360: **Function** SysUtils StrToInt64 Convert an integer **string** **into** an Int64 value
 24361: **Function** SysUtils StrToInt64Def Convert a **string** **into** an Int64 value **with default**
 24362: **Function** SysUtils StrToIntDef Convert a **string** **into** an Integer value **with default**
 24363: **Function** SysUtils StrToTime Converts a time **string** **into** a TDateTime value
 24364: **Function** StrUtils StuffString Replaces a part **of** one **string** **with** another
 24365: **Function** System Succ Increment an ordinal variable
 24366: **Function** Math Sum Return the sum **of** an **array** **of** floating point values
 24367:
 24368: T
 24369: **Function** Math Tan The Tangent **of** a number
 24370: **Type** Classes TBits An **object** that can hold an infinite number **of** Boolean values
 24371: Variable ConvUtils TConvFamily Defines a family **of** measurement types **as** used by Convert
 24372: **Type** ConvUtils TConvType Defines a measurement **type** **as** used by Convert
 24373: **Type** System TDateTime Data **type** holding a date **and** time value
 24374: **Type** System Text Defines a file **as** a text **file**
 24375: **Type** System TextFile Declares a file **type** **for** storing lines **of** text
 24376: **Type** SysUtils TFloatFormat Formats **for** use **in** floating point number display functions
 24377: **Type** SysUtils TFormatSettings A record **for** holding locale values **for** thread-safe functions
 24378: Keyword **Then** Part **of** an if statement - starts the true clause
 24379: Variable SysUtils ThousandSeparator The character used **to** display the thousands separator
 24380: Keyword **ThreadVar** Defines variables that are given separate instances per thread
 24381: **Function** SysUtils Time Gives the current time
 24382: Variable SysUtils TimeAMString Determines AM value **in** DateTimeToString **procedure**
 24383: Variable SysUtils TimePMString Determines PM value **in** DateTimeToString **procedure**
 24384: Variable SysUtils TimeSeparator The character used **to** separate display time fields
 24385: **Function** SysUtils TimeToStr Converts a TDateTime time value **to** a **string**
 24386: **Type** Classes TList General purpose container **of** a list **of** objects
 24387: Keyword **To** Prefixes an incremental for loop target value
 24388: **Type** System TObject The base class **type** that is ancestor **to** all other classes
 24389: **Function** DateUtils Tomorrow Gives the date tomorrow
 24390: **Type** Dialogs TOpenDialog Displays a file selection dialog
 24391: **Type** Types TPoint Holds X **and** Y integer values
 24392: **Type** Dialogs TPrintDialog Class that creates a printer selection **and** control dialog
 24393: **Type** Types TRect Holds rectangle coordinate values
 24394: **Type** SysUtils TReplaceFlags Defines options **for** the StringReplace routine
 24395: **Function** SysUtils Trim Removes leading **and** trailing blanks **from** a **string**
 24396: **Function** SysUtils TrimLeft Removes leading blanks **from** a **string**
 24397: **Function** SysUtils TrimRight Removes trailing blanks **from** a **string**
 24398: **Function** System Trunc The integer part **of** a floating point number
 24399: **Procedure** System Truncate Truncates a file size - removes all data after the current position
 24400: Keyword **Try** Starts code that has error trapping
 24401: **Type** Dialogs TSaveDialog Displays a dialog **for** selecting a save file name
 24402: **Type** SysUtils TSearchRec Record used **to** hold data **for** FindFirst **and** FindNext
 24403: **Type** Classes TStringList Holds a variable length list **of** strings
 24404: **Type** SysUtils TSysCharSet Characters used **by** supplied string parsing functions
 24405: **Type** System TThreadFunc Defines the function **to** be called by BeginThread
 24406: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold **for** 2 digit year **string** conversions
 24407: Keyword **Type** Defines a new category **of** variable **or** process
 24408:
 24409: U
 24410: Compiler Directive \$UnDef Undefines a compiler directive symbol - **as** used by IfDef
 24411: Keyword **Unit** Defines the start **of** a unit file - a Delphi module
 24412: Keyword **Until** Ends a Repeat control loop
 24413: **Function** System UpCase Convert a Char value **to** upper case
 24414: **Function** SysUtils UpperCase Change lower case characters **in** a **string** **to** upper case
 24415: Keyword **Uses** Declares a list **of** Units **to** be imported
 24416:
 24417: V
 24418: **Procedure** System Val Converts number strings **to** integer **and** floating point values

```

24419: Keyword Var Starts the definition of a section of data variables
24420: Type System Variant A variable type that can hold changing data types
24421: Function Variants VarType Gives the current type of a Variant variable
24422: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
24423: Directive Virtual Allows a class method to be overriden in derived classes
24424:
24425: W
24426: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
24427: Keyword While Repeat statements whilst a continuation condition is met
24428: Type System WideChar Variable type holding a single International character
24429: Function System WideCharToString Copies a null terminated WideChar string to a normal string
24430: Type System WideString A data type that holds a string of WideChars
24431: Keyword With A means of simplifying references to structured variables
24432: Type System Word An integer type supporting values 0 to 65535
24433: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
24434: Procedure System Write Write data to a binary or text file
24435: Procedure System WriteLn Write a complete line of data to a text file
24436:
24437: X
24438: Compiler Directive $X Controls some Pascal extension handling
24439: Keyword Xor Boolean Xor or bitwise Xor of two arguments
24440: Y
24441: Compiler Directive $Y Determines whether application symbol information is built
24442: Function DateUtils Yesterday Gives the date yesterday
24443:
24444: Z
24445: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
24446:
24447:
24448:
24449: procedure SIRegister_uPSUtils(CL: TPPascalCompiler);
24450: begin //'TPSBaseType', '').SetString( Byte);
24451: PSMainProcName', 'String').SetString( '!MAIN');
24452: PSMainProcNameOrg', 'String').SetString( 'Main Proc');
24453: 'PSLowBuildSupport', 'LongInt').SetInt( 12);
24454: CL.AddConstantN('PSCurrentBuildNo', 'LongInt').SetInt( 23);
24455: 'PSCurrentversion', 'String').SetString( '1.31');
24456: 'PSValidHeader', 'LongInt').SetInt( 1397769801);
24457: 'PSAddrStackStart', 'LongInt').SetInt( 1610612736);
24458: 'PSAddrNegativeStackStart', 'LongInt').SetInt( 1073741824);
24459: 'btReturnAddress', 'LongInt').SetInt( 0);
24460: 'btU8', 'LongInt').SetInt( 1);
24461: 'btS8', 'LongInt').SetInt( 2);
24462: 'btU16', 'LongInt').SetInt( 3);
24463: 'btS16', 'LongInt').SetInt( 4);
24464: 'btU32', 'LongInt').SetInt( 5);
24465: 'btS32', 'LongInt').SetInt( 6);
24466: 'btSingle', 'LongInt').SetInt( 7);
24467: 'btDouble', 'LongInt').SetInt( 8);
24468: 'btExtended', 'LongInt').SetInt( 9);
24469: 'btString', 'LongInt').SetInt( 10);
24470: 'btRecord', 'LongInt').SetInt( 11);
24471: 'btArray', 'LongInt').SetInt( 12);
24472: 'btPointer', 'LongInt').SetInt( 13);
24473: 'btPChar', 'LongInt').SetInt( 14);
24474: 'btResourcePointer', 'LongInt').SetInt( 15);
24475: 'btVariant', 'LongInt').SetInt( 16);
24476: 'btS64', 'LongInt').SetInt( 17);
24477: 'btU64', 'LongInt').SetInt( 30);
24478: 'btChar', 'LongInt').SetInt( 18);
24479: 'btWideString', 'LongInt').SetInt( 19);
24480: 'btWideChar', 'LongInt').SetInt( 20);
24481: 'btProcPtr', 'LongInt').SetInt( 21);
24482: 'btStaticArray', 'LongInt').SetInt( 22);
24483: 'btSet', 'LongInt').SetInt( 23);
24484: 'btCurrency', 'LongInt').SetInt( 24);
24485: 'btClass', 'LongInt').SetInt( 25);
24486: 'btInterface', 'LongInt').SetInt( 26);
24487: 'btNotificationVariant', 'LongInt').SetInt( 27);
24488: 'btUnicodeString', 'LongInt').SetInt( 28);
24489: 'btType', 'LongInt').SetInt( 130);
24490: 'btEnum', 'LongInt').SetInt( 129);
24491: 'btExtClass', 'LongInt').SetInt( 131);
24492: 'CM_A', 'LongInt').SetInt( 0);
24493: 'CM_CA', 'LongInt').SetInt( 1);
24494: 'CM_P', 'LongInt').SetInt( 2);
24495: 'CM_pv', 'LongInt').SetInt( 3);
24496: 'CM_PO', 'LongInt').SetInt( 4);
24497: 'Cm_C', 'LongInt').SetInt( 5);
24498: 'Cm_G', 'LongInt').SetInt( 6);
24499: 'Cm(CG)', 'LongInt').SetInt( 7);
24500: 'Cm_CNG', 'LongInt').SetInt( 8);
24501: 'Cm_R', 'LongInt').SetInt( 9);
24502: 'Cm_ST', 'LongInt').SetInt( 10);
24503: 'Cm_Pt', 'LongInt').SetInt( 11);
24504: 'CM_CO', 'LongInt').SetInt( 12);
24505: 'Cm_cv', 'LongInt').SetInt( 13);
24506: 'cm_sp', 'LongInt').SetInt( 14);
24507: 'cm_bn', 'LongInt').SetInt( 15);

```

```

24508: 'cm_vm', 'LongInt').SetInt( 16);
24509: 'cm_sf','LongInt').SetInt( 17);
24510: 'cm_fg','LongInt').SetInt( 18);
24511: 'cm_puevh','LongInt').SetInt( 19);
24512: 'cm_poexh','LongInt').SetInt( 20);
24513: 'cm_in','LongInt').SetInt( 21);
24514: 'cm_spc','LongInt').SetInt( 22);
24515: 'cm_inc','LongInt').SetInt( 23);
24516: 'cm_dec','LongInt').SetInt( 24);
24517: 'cm_nop','LongInt').SetInt( 255);
24518: 'Cm_PG','LongInt').SetInt( 25);
24519: 'Cm_P2G','LongInt').SetInt( 26);
24520: CL.AddTypeS('TbtU8', 'Byte');
24521: CL.AddTypeS('TbtS8', 'ShortInt');
24522: CL.AddTypeS('TbtU16', 'Word');
24523: CL.AddTypeS('TbtS16', 'SmallInt');
24524: CL.AddTypeS('TbtU32', 'Cardinal');
24525: CL.AddTypeS('TbtS32', 'Longint');
24526: CL.AddTypeS('TbtSingle', 'Single');
24527: CL.AddTypeS('TbtDouble', 'double');
24528: CL.AddTypeS('TbtExtended', 'Extended');
24529: CL.AddTypeS('tbtCurrency', 'Currency');
24530: CL.AddTypeS('tbts64', 'int64');
24531: CL.AddTypeS('Tbtu64', 'uint64');
24532: CL.AddTypeS('TbtString', 'string');
24533: Function MakeHash( const s : TbtString ) : Longint;
24534: //   TbtString = {$IFDEF DELPHI2009UP}AnsiString{$ELSE}String{$ENDIF};
24535: // 'PointerSize','LongInt').SetInt( IPointer ( 8 4 ) );
24536: end;
24537:
24538: -----
24539: mapX:
24540: if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
24541:   writeln('cologne map found');
24542:   GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
24543:   writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
24544:   OpenMapX('church trier');
24545:   GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
24546:   writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));
24547:   >>> //latitude: '50.94133705' longitude: '6.9581207610076'
24548:   // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
24549:   CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TPos): string;
24550:   Function SendInput( cInputs : UINT; var pInputs : TInput; cbSize : Integer ) : UINT';
24551:   Function GetLastInputInfo( var plii : TLastInputInfo ) : BOOL';
24552:
24553:
24554: maxbox Ref:
24555: Signature:
24556: SHA1: maxbox3.exe A5C1A64E94EBC01D01B67DB35C3E27E50607DFFB
24557: CRC32: maxbox3.exe 37CC9E45
24558:
24559: Ref:
24560: 1. writeln(SHA1(Exepath+'\maxbox3.exe'))
24561: 2. shdig: TSHA1Digest;
24562:   shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
24563:   for i:= 0 to 19 do write(Bytetohex(shdig[i]));
24564:
24565: 3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
24566:
24567:
24568: https://www.virustotal.
com/en/file/fbcae4b521df1f1899846398b07f8f6f99ce329262b93860aa84f83a6222d326/analysis/1426370468/
24569:
24570: VirusTotal metadata
24571: -----
24572: First submission 2015-03-13 17:03:29 UTC ( 16 hours, 48 minutes ago )
24573: Last submission 2015-03-14 08:15:56 UTC ( 1 hour, 35 minutes ago )
24574: File names Surprise mx4
24575: maxbox3.exe
24576: maxbox3_9.exe
24577:
24578: SHA256: fbcae4b521df1f1899846398b07f8f6f99ce329262b93860aa84f83a6222d326
24579: File name: maxbox3.exe
24580: Detection ratio: 0 / 57
24581: 2015-03-14 22:15:56 UTC ( 1 hour, 44 minutes ago )
24582:
24583: MD5 7ce0677a94d70b7dc67526465260ba87
24584: SHA1 a5cla64e94ebc0ld01b67db35c3e27e50607dff8
24585: SHA256 fbcae4b521df1f1899846398b07f8f6f99ce329262b93860aa84f83a6222d326
24586: ssdeep 393216:610lPRgJGfw/5Cc6KCj2KCK0rUfETSmH7tYWuIbe:6cPOJGtf2bqUr
24587: authentihash cac578354f8b36dc0f55991141333c732b866244b48276a75206035105b6e318
24588: imphash 0c55892237c705323e5d0aeee689ce0e
24589: File size 23.4 MB ( 24496640 bytes )
24590: File type Win32 EXE
24591: Magic literal
24592: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
24593:
24594: TrID Windows ActiveX control (36.4%)
24595: Inno Setup installer (34.3%)

```

```
24596:     InstallShield setup (13.4%)
24597:     Win32 EXE PECompact compressed (generic) (13.0%)
24598:     Win32 Executable (generic) (1.4%)
24599:
24600:
24601:     ExifTool file metadata
24602: -----
24603:
24604:     SpecialBuild mx4
24605:     LegalTrademarks maxbox
24606:     SubsystemVersion 4.0
24607:     Comments reduce to the max
24608:     LinkerVersion 2.25
24609:     ImageVersion 0.0
24610:     FileSubtype 0
24611:     FileVersionNumber 3.9.9.190
24612:     LanguageCode German (Swiss)
24613:     FileFlagsMask 0x003f
24614:     FileDescription maxbox Delphi VM
24615:     CharacterSet Windows, Latin1
24616:     InitializedDataSize 5273600
24617:     FileOS Win32
24618:    TimeStamp 2015:03:14 13:42:07+01:00
24619:     MIMEType application/octet-stream
24620:     LegalCopyright Free Pascal Script
24621:     FileVersion 3.9.9.190
24622:     SpeziellesBuild mx4 Compiler Engine
24623:     FileType Win32 EXE
24624:     PEType PE32
24625:     InternalName Surprise mx4
24626:     ProductVersion 3.9 Solar mx4
24627:     UninitializedDataSize 0
24628:     OSVersion 4.0
24629:     OriginalFilename maxbox3_9.exe
24630:     Subsystem Windows GUI
24631:     MachineType Intel 386 or later, and compatibles
24632:     CompanyName kleiner kommunikation
24633:
24634:     CodeSize 19222016
24635:     ProductName maxbox
24636:     ProductVersionNumber 3.9.9.190
24637:     EntryPoint 0x1255fb4
24638:     ObjectFileType Executable application
24639:
24640:     ExifTool file metadata
24641: -----
24642:     Developer metadata
24643:     Publisher kleiner kommunikation
24644:     Product maxbox
24645:     Original name maxbox3_9.exe
24646:     Internal name Surprise mx4
24647:     File version 3.9.9.190
24648:     Description maxbox Delphi VM
24649:     Comments reduce to the max
24650:
24651:     PE header basic information
24652:     Target machine Intel 386 or later processors and compatible processors
24653:     Compilation timestamp 2015-03-14 12:42:07
24654:     Entry Point 0x01255FB4
24655:     Number of sections 10
24656:
24657:     10 PE sections
24658:
24659:     Name Virtual address Virtual size Raw size Entropy MD5
24660:     .text 4096 19172284 19172352 6.60 75728b866e3bcd08cf86e375df33c9b7
24661:     .idata 19177472 49400 49664 6.56 8994d7f351a357c3ebbe81be2f4b09db
24662:     .data 19230720 2677516 5.60 5b5596962bc0b70c31ee9c27d4624115
24663:     .bss 19501056 381596 0 0.00 d41d8cd98f00b204e9800998ecf8427e
24664:     .idata 19886080 56334 56832 5.63 52b428a4d181aa8c820b23fde28378d2
24665:     .edata 19943424 77 512 0.89 fb4d3157fedb6b411b16928354a0a9a7
24666:     .tls 19947520 208 0 0.00 d41d8cd98f00b204e9800998ecf8427e
24667:     .rdata 19951616 24 512 0.27 7621ca56bcd301aleba785f758f613a4
24668:     .reloc 19955712 1223052 1223168 6.76 8d68bfb42a70b18cad2ee914d182e608
24669:     .rsrc 21180416 3724800 3724800 5.36 7e1c1b6d6be0da46b4ad2b19b12061a
24670:
24671:     PE imports
24672:     PE exports CreateIncome
24673:
24674:     Number of PE resources by type
24675:     RT_BITMAP 839
24676:     RT_STRING 307
24677:     RT_RCDATA 163
24678:     RT_ICON 55
24679:     RT_GROUP_ICON 44
24680:     RT_CURSOR 33
24681:     RT_GROUP_CURSOR 28
24682:     WAVE 9
24683:     RT_DIALOG 2
24684:     RT_HTML 2
```

```

24685: RT_MESSAGEGETABLE 1
24686: RT_MANIFEST 1
24687: AVI 1
24688: RT_VERSION 1
24689:
24690: PE imports
24691: [+] AVICAP32.DLL [+] AVICAP32.dll [+] GLU32.dll
24692: [+] IMAGEHELP.DLL [+] MSVCRT.DLL [+] MSVFW32.DLL
24693: [+] SHFolder.dll [+] URLMON.DLL
24694: [+] advapi32.dll [+] comct132.dll [+] comdlg32.dll
24695: [+] gdi32.dll [+] imagehlp.dll [+] imm32.dll
24696: [+] iphpapi.dll [+] kernel32.dll [+] mpr.dll
24697: [+] msacm32.dll [+] ole32.dll [+] oleacc.dll
24698: [+] oleaut32.dll [+] oledlg.dll [+] opengl32.dll
24699: [+] shell32.dll [+] shlwapi.dll [+] user32.dll
24700: [+] usp10.dll [+] version.dll [+] winhttp.dll
24701: [+] wininet.dll [+] winmm.dll [+] winspool.drv
24702: [+] ws2_32.dll [+] wssock32.dll [+] msimg32.dll
24703:
24704: Ref:
24705: https://www.virustotal.com/en/file/fbcae4b521df1f1899846398b07f8f6f99ce329262b93860aa84f83a6222d326/analysis/1426370468/
24706:
24707: ****
24708: Release Notes maxbox 3.9.9.190 April 2015 CODEsign
24709: ****
24710: Add 16 Units, 1 Tutor, SOAPConn, AVI Res, OLEUtils, Wave ACM
24711: Refactor DB Constructor - Virtual Constructor - Override
24712:
24713: 1085 unit uPSI_JvAnimate //JCL
24714: 1086 unit uPSI_DBXCharDecoder; //DBX
24715: 1087 unit uPSI_JvDBLists; //JCL
24716: 1088 unit uPSI_JvFileInfo; //JCL
24717: 1089 unit uPSI_SOAPConn; //VCL
24718: 1090 unit uPSI_SOAPLinked; //VCL
24719: 1091 unit uPSI_XSBuiltIns; //VCL
24720: 1092 unit uPSI_JvgDigits; //JCL
24721: 1093 unit uPSI_JvDesignUtils;
24722: 1094 unit uPSI_JvgCrossTable;
24723: 1095 unit uPSI_JvgReport; 1096 unit uPSI_JvDBRichEdit;
24724: 1097 unit uPSI_JvWinHelp;
24725: 1098 unit uPSI_WaveConverter; 1099 unit uPSI_ACMConvertor;
24726: 1100 unit XSBuiltIns_Routines
24727:
24728:
24729: SHA1: maxbox3.exe A5C1A64E94EBC01D01B67DB35C3E27E50607DFFB
24730: CRC32: maxbox3.exe 37CC9E45
24731:
24732: ****
24733: Release Notes maxbox 3.9.9.180 Feb 2015 CODEsign
24734: ****
24735: Add 20 Units, 1 Slide, Tutor, Big Numbers, TestFramework, GEOInfo
24736:
24737: 1065 unit uPSI_UDict; //DFF
24738: 1066 unit uPSI_ubigFloatV3; //DFF
24739: 1067 unit uPSI_UBigIntsV4; //DFF
24740: 1068 unit uPSI_ServiceMgr2; //mX
24741: 1069 unit uPSI_UP10Build; //PS
24742: 1070 unit uPSI_UParser10; //PS
24743: 1071 unit uPSI_IdModBusServer; //MB
24744: 1072 unit uPSI_IdModBusClient; +MBUtils //MB
24745: 1073 unit uPSI_ColorGrd; //VCL
24746: 1074 unit uPSI_DirOutln; //VCL
24747: 1075 unit uPSI_Gauges; //VCL
24748: 1076 unit uPSI_CustomizeDlg; //VCL
24749: 1077 unit uPSI_ActnMan; //VCL
24750: 1078 unit uPSI_CollPanl; //VCL
24751: 1079 unit uPSI_Calendar2; //VCL
24752: 1080 unit uPSI_IBCCtrls; //VCL
24753: 1081 unit uPSI_IdStackWindows; //Indy
24754: 1082 unit uPSI_CTSVendorUtils;
24755: 1083 unit uPSI_VendorTestFramework;
24756: 1084 unit uPSI_TInterval;
24757:
24758: SHA1: maxbox3.exe 3D7F88BE9687CB834A5E2DAED08B23358484FEC0
24759: CRC32: maxbox3.exe E2ADE828
24760:
24761: ****
24762: Release Notes maxbox 3.9.9.160 January 2015 CODEsign
24763: ****
24764: Add 9 Units, 2 Slides 1 Tutor, CLXUp, ExampleEdition, UnitConverter
24765: ExecuteProcess (MultiProcessor), ConsoleCapture (DOS)
24766:
24767: 1053 unit uPSI_BigIni //Hinzen
24768: 1054 unit uPSI_ShellCtrls; //VCL
24769: 1055 unit uPSI_fMath; //FMath
24770: 1056 unit uPSI_fComp; //FMath
24771: 1057 unit uPSI_HighResTimer; //Lauer
24772: 1058 unit uconvMain; (Unit Converter) //PS

```

```
24773: 1059 unit uPSI_uconvMain; //PS
24774: 1060 unit uPSI_ParserUtils; //PS
24775: 1061 unit uPSI_UPSUtils; //PS
24776: 1062 unit uPSI_ParserU; //PS
24777: 1063 unit uPSI_TypeInfo; //VCL
24778: 1064 unit uPSI_ServiceMgr; //mX
24779:
24780: ****
24781: Release Notes maXbox 3.9.9.120 December 2014 CODEsign
24782: ****
24783: Add 10 Units, 1Slides, NeuralNetwork, Pan3D View, GDIBackend
24784:
24785: 1043 unit uPSI_NeuralNetwork;
24786: 1044 unit uPSI_StExpr;
24787: 1045 unit uPSI_GR32_Geometry;
24788: 1046 unit uPSI_GR32_Containers;
24789: 1047 unit uPSI_GR32_Backends_VCL,
24790: 1048 unit uPSI_StSaturn; //Venus+Pluto+Mars+Merc+JupSat++
24791: 1049 unit uPSI_JclParseUses;
24792: 1050 unit uPSI_JvFinalize;
24793: 1051 unit uPSI_panUnit1;
24794: 1052 unit uPSI_DD83ul; //Arduino Tester
24795:
24796:
24797: Published Doc maXbox Tutors Starter Introduction 2014
24798: Tutorial 00 Blix the Programmer
24799: Tutorial 01 Procedural-Coding
24800: Tutorial 02 OO-Coding
24801: Tutorial 03 Modular Coding
24802: Tutorial 04 UML Coding
24803: Tutorial 05 Internet Coding
24804: Tutorial 06 Network Coding
24805: Tutorial 07 Game Coding
24806: Tutorial 08 Operating System Coding
24807: Tutorial 09 Database Coding
24808: Tutorial 10 Statistic Coding
24809: Tutorial 11 Forms Coding
24810: Tutorial 12 SQL Coding
24811: Tutorial 13 Crypto Coding
24812: Tutorial 14 Parallel Coding
24813: Tutorial 15 Serial Coding
24814: Tutorial 16 Event Driven Coding
24815: Tutorial 17 Web Server Coding
24816: Tutorial 18 Arduino Coding and Web of Things
24817: Tutorial 18_3 Arduino RGB LED Breadboard and Source LED Zip
24818: Tutorial 19 WinCOM /Arduino Coding and Source LED COM
24819: Tutorial 20_1 Regular Expressions V2
24820: Tutorial 21 Android av. End of 2014 and Basta LED Things and Code ADK SeekBar
24821: Tutorial 22 Services Coding
24822: Tutorial 23 Real Time Code
24823: Tutorial 24 Clean Code
24824: Tutorial 25 Configuration
24825: Tutorial 26 TCP Sockets
24826: Tutorial 27 XML Coding
24827: Tutorial 28 DLL Coding
24828: Tutorial 29 UML Modeling
24829: Tutorial 30 WOT Web of Things and Basta 2014 Arduino and maXbox
24830: Tutorial 31 Closures
24831: Tutorial 32 SQL Server Firebird
24832: Tutorial 33 Oscilloscope
24833: Tutorial 34 GPS Codes
24834: Tutorial 35 Web Box
24835: Tutorial 39 Maps Coding
24836: Tutorial 39 Maps2 Coding
24837: http://www.slideshare.net/maxkleiner1/codesign-2015
24838:
24839:
24840: Published Doc maXbox Example Edition 2015
24841: \example_edition\01_Algorithm';
24842: \example_edition\02_Graphics';
24843: \example_edition\03_Games';
24844: \example_edition\04_Multimedia';
24845: \example_edition\05_Internet';
24846: \example_edition\06_Communication';
24847: \example_edition\07_Geographical';
24848: \example_edition\08_Operating';
24849: \example_edition\09_Database';
24850: \example_edition\10_Science';
24851: \example_edition\11_EMBEDDED';
24852: \example_edition\12_Security';
24853: \example_edition\13_General';
24854: \example_edition\13_General';
24855:
24856: ref:
24857: SHA1: maXbox3.exe A5C1A64E94EBC01D01B67DB35C3E27E50607DFFB
24858: CRC32: maXbox3.exe 37CC9E45
24859:
24860: ---- bigbitbox code_cleared_checked----
```